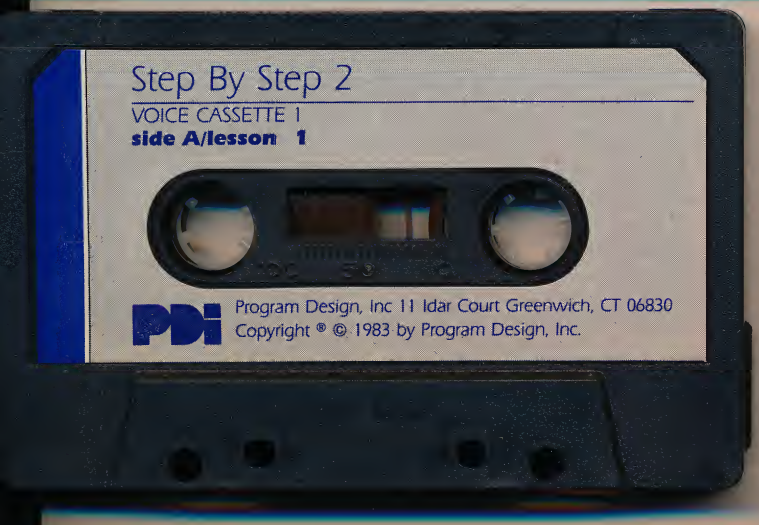
# STEP BY STEP TWO

## An Intermediate Course in BASIC Programming

Program Design, Inc. 11 Idar Court Greenwich, CT 06830

Step By Step 2
VOICE CASSETTE 1
side B/lesson 2
PDI
Program Design, Inc 11 Idar Court Greenwich, CT 06830
Copyright ® © 1983 by Program Design, Inc.

Step By Step 2
VOICE CASSETTE 2
side A/lesson 3, part 1
PDI
Program Design, Inc 11 Idar Court Greenwich, CT 06830
Copyright ® © 1983 by Program Design, Inc.


Step By Step 2
VOICE CASSETTE 1
side A/lesson 1
PDI
Program Design, Inc 11 Idar Court Greenwich, CT 06830
Copyright ® © 1983 by Program Design, Inc.

Step By Step 2
VOICE CASSETTE 2
side B/lesson 3, part 2
PDI
Program Design, Inc 11 Idar Court Greenwich, CT 06830
Copyright ® © 1983 by Program Design, Inc.

Step By Step 2

VOICE CASSETTE 3

**side B**

PDI Program Design, Inc 11 Idar Court Greenwich, CT 06830

Copyright ℗ © 1982 by Program Design, Inc.

Step By Step 2

VOICE CASSETTE 4

**side A/lesson 5, part 1**

PDI Program Design, Inc 11 Idar Court Greenwich, CT 06830

Copyright ® © 1983 by Program Design, Inc.

Step By Step 2

VOICE CASSETTE 3

**side A/lesson 4, part 1**

PDI Program Design, Inc 11 Idar Court Greenwich, CT 06830

Copyright ® © 1983 by Program Design, Inc.

Step By Step 2

VOICE CASSETTE 4

**side B/lesson 5, part 2**

PDI Program Design, Inc 11 Idar Court Greenwich, CT 06830

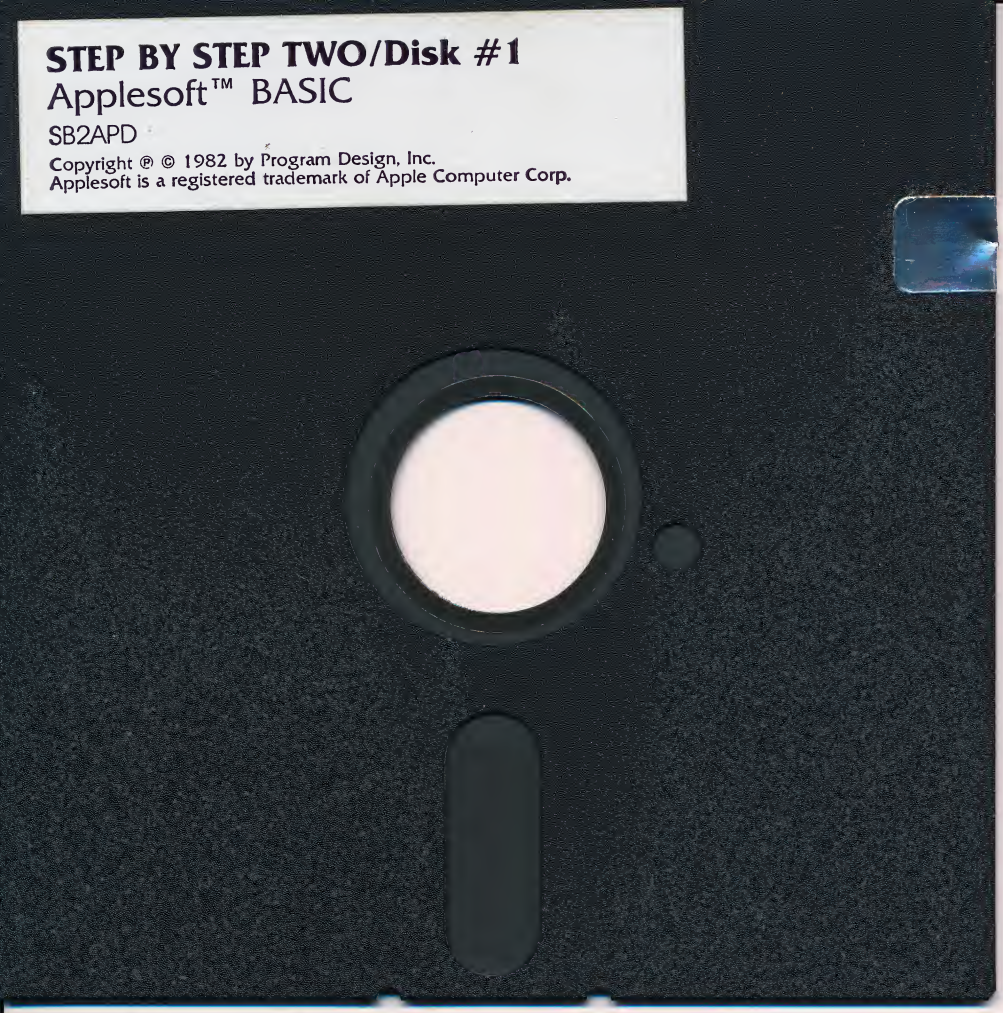
Copyright ® © 1983 by Program Design, Inc.

**STEP BY STEP TWO/Disk #1**

Applesoft™ BASIC

SB2APD

Copyright ® © 1982 by Program Design, Inc.
Applesoft is a registered trademark of Apple Computer Corp.



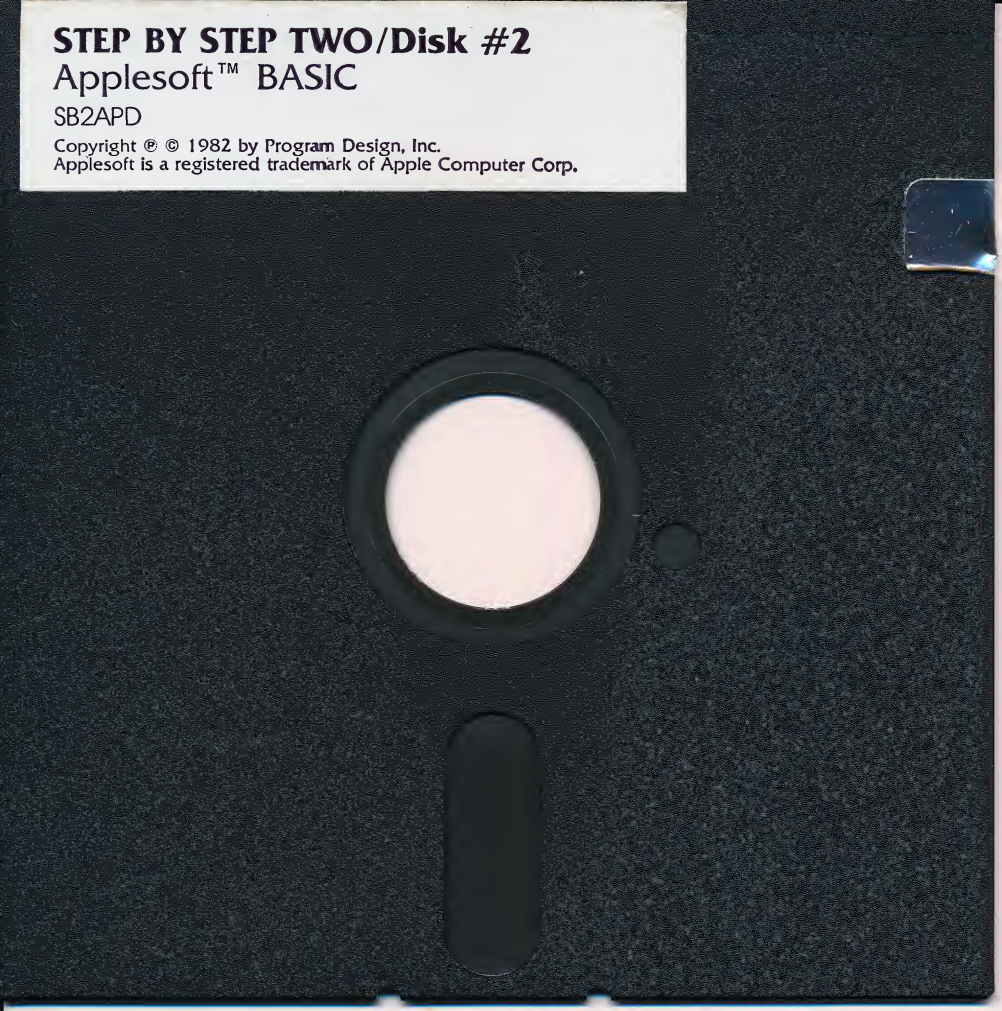
ALLENBACH
INDUSTRIES, INC.

**STEP BY STEP TWO/Disk #2**

Applesoft™ BASIC

SB2APD

Copyright ℗ © 1982 by Program Design, Inc.
Applesoft is a registered trademark of Apple Computer Corp.



ALLENBACH
INDUSTRIES, INC.

# STEP BY STEP TWO

## An Intermediate Course in BASIC Programming

Program Design, Inc. 95 East Putnam Avenue Greenwich, CT 06830

# STEP BY STEP TWO

## An Intermediate Course in BASIC Programming

Program Design, Inc., 95 East Putnam Avenue
Greenwich, CT 06830 203–661–8799

**Project Designer:** JOHN VICTOR

**Programmers:** STEPHEN CHMIELEWSKI
KATHLEEN FORTMEIER

**Editor:** LYN SANDOW

**Graphic Design:** HOWARD PETLACK

Copyright © 1983 by Program Design, Inc.
All rights reserved.

No part of this publication may be reproduced
in whole or in part, or stored in a retrieval system, or transmitted
in any form or by any means, electronic, mechanical,
photocopying or recording, without written permission of the publisher.
For information relating to permissions,
write to Program Design, Inc.
95 East Putnam Avenue, Greenwich, CT 06830

First Edition

Printed in U.S.A.

Apple™ is a trademark of Apple Computer Corp.

## Table of Contents


| | |
|---|---|
| Introduction | 5 |
| Where Will I Find The Specific Material I Need? | 7 |
| Quizzes | 8 |
| Answers to Quizzes | 26 |
| Lesson 1, Part 1–Summary | 27 |
| Lesson 1, Part 1–Practice Problems | 28 |
| Lesson 1, Part 2–Summary | 32 |
| Lesson 1, Part 2–Practice Problems | 32 |
| Lesson 2, Part 1–Summary | 36 |
| Lesson 2, Part 1–Practice Problems | 36 |
| Lesson 2, Part 2–Summary | 39 |
| Lesson 2, Part 2–Practice Problems | 39 |
| Lesson 3, Part 1–Summary | 43 |
| Lesson 3, Part 1–Practice Problems | 44 |
| Lesson 3, Part 2–Summary | 48 |
| Lesson 3, Part 2–Practice Problems | 49 |
| Lesson 4, Part 1–Summary | 52 |
| Lesson 4, Part 1–Practice Problems | 53 |
| Lesson 4, Part 2–Summary | 57 |
| Lesson 4, Part 2–Practice Problems | 58 |
| Lesson 5, Part 1–Summary | 62 |
| Lesson 5, Part 1–Practice Problems | 64 |
| Lesson 5, Part 2–Summary | 68 |
| Lesson 5, Part 2–Practice Problems | 69 |
| Supplementary Practice | 73 |
| Exhibits | 88 |
| Progress Charts | 95 |

## WHAT IS STEP BY STEP TWO?

*Step By Step Two, An Intermediate Course in BASIC,* is a continuation of *New Step By Step.* The original course, introduced in 1981, has become the industry standard for learning beginning BASIC. Both courses employ advanced techniques for introducing, reinforcing and testing material to be learned. The student works through each lesson on the computer; a voice accompaniment allows him or her to devote full attention to the screen display. At the completion of each lesson segment the student goes to the workbook. The workbook presents a summary of the material covered in the lesson, plus practice problems that reinforce and expand what was taught in the lesson. Each of the five lessons consists of two parts. Each lesson is followed by a quiz that enables the student to test his or her progress.

## WHAT DOES STEP BY STEP TWO TEACH?

The following skills, vital for proficient programming in BASIC, are included:

| | |
|---|---|
| **PEEK** and **POKE** | Control characters |
| Default values | Hexadecimal and binary numbers |
| Memory map | Machine monitor |
| **CALL** | String logic |
| ASCII codes | String arrays |
| **CHR$** and **ASC** functions | High-resolution graphics |
| RAM and ROM | Screen memory |

Additional sections, titled *Supplementary Practice: An Introduction to Game Design* and *Music for Your APPLE II*, are included in this workbook. This material is designed for people who have successfully completed *Step By Step Two* and would like more practice.

## HOW WILL LEARNING BE MOST EFFECTIVE?

1. You *cannot* learn to program without actually writing programs. This course will be of little value unless you actually practice what you learn in each lesson.
2. The lessons in the course should be completed in sequence. No lessons should be skipped, even if you feel that you already know the material being covered.
3. It is not a good idea to do too much in a session. One lesson part is probably the limit for most people, although some will want to do less.
4. It is best not to let too much time elapse between sessions. Short daily sessions are better than long weekly sessions.

5. You may need to repeat lessons to insure maximum learning.
6. The practice sessions are extremely important to effective learning. You should try *all* of the material in the practice session.
7. A good way to learn programming is to take someone else's program, make your own modifications to the program and observe the results. This should be done with all the programs given in the practice sessions.

## HOW SHOULD I BEGIN?

1. You are now ready to begin *Step By Step Two*. If you have an APPLE II Plus, insert Disk One into disk drive #1 and turn on the computer. If you have an APPLE II, push the CTRL key and the B key at the same time. Then type **PR#6.**

   A menu will appear on the screen. Select LESSON 1 PART 1.
2. Load Cassette 1, Side A into a regular cassette player; the unit is not connected to your APPLE II. The narrator will tell you how and when to advance the screen visuals and/or stop the player to allow you time to answer questions.
3. Should you wish to stop a lesson at some point before it is completed, hold down the CTRL key and C at the same time. If you wish to start the lesson again, type **RUN** followed by the lesson name, then press RETURN. The voice cassette will have to be rewound to the correct spot to be rerun.

   Should you wish to return to the menu, type **RUN HELLO** and press RETURN. The menu will appear on the screen.
4. As you complete each lesson segment, fill in your progress on the chart located at the back of this workbook. Several copies of the chart are included, should the course be used by more than one person.
5. After both parts of each lesson are completed, take the quiz for that lesson. The quizzes are located on the disks. To load a quiz from "ground 0," go to the menu and make your selection.

   Another way to call up a quiz is to type RUN followed by the quiz name, then press RETURN.

## WHERE WILL I FIND THE SPECIFIC MATERIAL I NEED?

*Step By Step Two* is presented on two disks:

DISK #1

```
(1) LESSON 1 PART 1
(2) LESSON 1 PART 2
(3) QUIZ 1
(4) LESSON 2 PART 1
(5) LESSON 2 PART 2
(6) QUIZ 2
(7) LESSON 3 PART 1
(8) LESSON 3 PART 2
(9) QUIZ 3
```

DISK #2

```
(1) LESSON 4 PART 1
(2) LESSON 4 PART 2
(3) QUIZ 4
(4) LESSON 5 PART 1
(5) LESSON 5 PART 2
(6) QUIZ 5
```

The audio for *Step By Step Two* is presented on four cassettes:

**Cassette 1, Side A**
Lesson 1, Part 1
Lesson 1, Part 2

**Cassette 1, Side B**
Lesson 2, Part 1
Lesson 2, Part 2

**Cassette 2, Side A**
Lesson 3, Part 1

**Cassette 2, Side B**
Lesson 3, Part 2

**Cassette 3, Side A**
Lesson 4, Part 1

**Cassette 3, Side B**
Lesson 4, Part 2

**Cassette 4, Side A**
Lesson 5, Part 1

**Cassette 4, Side B**
Lesson 5, Part 2

**NOTE:** Because the disks in *Step By Step Two* are copy-protected and an overwhelming number of our customers request back-up disks, we have decided to include a set of back-up disks with each course. We suggest that you remove these disks from the kit and put them in a safe place until needed.

## QUIZZES

The quizzes are delivered via the computer. However, printed versions follow, should you wish them for group use.

### Quiz 1

```
                              QUESTION 1
WHICH OF THESE REPRESENTS THE ADDRESS
OF A MEMORY LOCATION IN THE COMPUTER?

(A) 32
(B) A(32)
(C) 32.33



                              QUESTION 2
A 32K COMPUTER CONTAINS ABOUT:

....... MEMORY LOCATIONS

(A)32
(B) 32ØØ
(C) 32ØØØ
(D) NONE OF THESE



                              QUESTION 3
THE LETTER  L  WOULD BE STORED IN
COMPUTER MEMORY AS A .......



                              QUESTION 4
TYPE A PRINT INSTRUCTION THAT WILL
GIVE YOU THE VALUE OF A NUMBER STORED
IN MEMORY LOCATION 5Ø.
```

QUESTION 5

TYPE THE INSTRUCTION WHICH WILL REPLACE THE VALUE IN MEMORY LOCATION 33 BY THE NUMBER 15.

QUESTION 6

```
10 FOR X = 1 TO 4
20 POKE 31 + X,X
30 NEXT X
40 WX = PEEK(33)
```

WHAT WILL BE THE VALUE OF WX IN THIS PROGRAM?

QUESTION 7

WHAT IS THE INSTRUCTION WHICH WILL ACTIVATE A MACHINE LANGUAGE PROGRAM WHICH STARTS IN MEMORY LOCATION 5?

QUESTION 8

TYPE IN AN INSTRUCTION WHICH ALLOWS NO PRINTING IN THE FIRST 8 COLUMNS OF THE SCREEN.

QUESTION 9

```
10 POKE 32,10
20 POKE 33,20
30 POKE 34,8
40 POKE 35,20
50 FOR X=1 TO 2
60 PRINT A$
70 NEXT X
```

WHAT WILL BE THE MAXIMUM LENGTH OF LINES PRINTED IN THIS PROGRAM?

## QUESTION 10

```
10 POKE 32,10
20 POKE 33,20
30 POKE 34,8
40 POKE 35,20
50 FOR X=1 TO 2
60 PRINT A$
70 NEXT X
```

WHAT IS THE TOTAL NUMBER OF LINES
THAT CAN APPEAR ON THE SCREEN AT ONE
ONE TIME?

## QUESTION 11

```
10 POKE 32,10
20 POKE 33,20
30 POKE 34,8
40 POKE 35,20
50 FOR X=1 TO 2
60 PRINT A$
70 NEXT X
```

WHAT COMMAND WOULD YOU USE TO RESET ALL
MARGINS TO THEIR DEFAULT VALUES?

## QUESTION 12

TYPE AN INSTRUCTION TO PRINT THE VALUE
OF MEMORY LOCATION 12000

## QUESTION 13

TYPE T IF THE STATEMENT IS TRUE;
F IF IT'S FALSE.

WHEN THE COMPUTER IS FIRST TURNED ON,
ALL THE MEMORY LOCATIONS CONTAIN 0'S.

QUESTION 14

WHICH COMMAND WILL PRINT THE VALUE
OF MEMORY LOCATION 48000?

(A) PRINT PEEK (48)
(B) PRINT PEEK (- 48000)
(C) PRINT PEEK (- 17536)

QUESTION 15

```
10 POKE -16368,0
20 A=PEEK(-16384)
30 IF A<127 THEN 20
40 FOR R=1 TO 10
45 M=PEEK(-16336)
50 NEXT R
60 GOTO 10
```

WHEN THIS PROGRAM IS RUN, THE VALUE
OF A:

(A) IS ALWAYS 0
(B) CHANGES EACH TIME A KEY IS PRESSED
(C) CANNOT BE DETERMINED HERE

QUESTION 16

```
10 POKE -16368,0
20 A=PEEK(-16384)
30 IF A<127 THEN 20
40 FOR R=1 TO 10
45 M=PEEK(-16336)
50 NEXT R
60 GOTO 10
```

WHICH LINE OF THE PROGRAM WOULD YOU
CHANGE TO VARY THE LOUDNESS OF THE
SOUND PRODUCED WHEN A KEY IS PRESSED?

## Quiz 2

QUESTION 1

USING CHR$, WRITE AN INSTRUCTION
TO PRINT THE LETTER B ON THE SCREEN.

QUESTION 2

```
10 FOR X=1 TO 10
20 ?X;CHR$(7);
30 NEXT X
```

WHEN THIS PROGRAM IS RUN, THE NUMBERS
FROM 1 TO 10 WILL BE PRINTED:

(A) ONE RIGHT UNDER THE OTHER
(B) IN A STRAIGHT LINE WITH A SPACE
    BETWEEN EACH ONE
(C) IN A LINE WITH A BELL RINGING EACH
    TIME A NUMBER APPEARS.

QUESTION 3

```
PRINT "HE SAID";CHR$(34);"YES";CHR$(34)
```

WHAT WILL THIS PRINT?

(A) HE SAID YES
(B)"HE SAID YES"
(C) HE SAID"YES"

QUESTION 4

```
X=ASC("A")    Y=ASC("ARM")    Z=ASC("AS")
```

X = Y = Z

TYPE T IF THE STATEMENT IS TRUE;
F IF IT'S FALSE.

QUESTION 5

```
10 PRINT"LETTER","ASCII"
20 PRINT
30 FOR X=65 TO 90
40 READ A$
.............................
60 NEXT X
70 DATA A,B,C,D,E,F,G.............
```

WRITE LINE 50 TO PRINT THE LETTERS
AND THEIR ASCII VALUES IN COLUMNS.

QUESTION 6

WHICH OF THESE INSTRUCTIONS WILL CHANGE
THE NUMBER IN A MEMORY LOCATION
ON THE APPLE?

(A) POKE 63000,20
(B) POKE 78000,10
(C) POKE 12134,70

QUESTION 7

WRITE AN INSTRUCTION THAT WILL SET
31000 AS THE HIGHEST ADDRESS A
BASIC PROGRAM CAN OCCUPY

QUESTION 8

TYPE T IF THE STATEMENT IS TRUE;
F IF IT'S FALSE.

A NUMBER STORED IN LOCATION 63000 IS
ERASED WHEN THE COMPUTER IS TURNED OFF.

QUESTION 9

```
10 FOR X=1024 TO 1027
20 READ A
30 POKE X,A
40 NEXT X
50 DATA 194,207,207,205
```

WHAT WILL THIS PROGRAM PRINT?

(A) BOOM **(in regular type)**
(B) BOOM **(in inverse type)**
(C) BOOM **(flashing type)**

## Quiz 3

QUESTION 1

WHICH OF THESE BINARY NUMBERS HAS THE LARGEST VALUE?

(A) ØØ1ØØØØ1
(B) ØØ1ØØØ1Ø
(C) ØØØ1111Ø

QUESTION 2

COMPLETE THIS ANALOGY

4 IS TO 8 AS:

(A) NYBBLE IS TO BYTE
(B) BIT IS TO BYTE
(C) BIT IS TO NYBBLE

QUESTION 3

ØØØØØ11

THIS NUMBER REPRESENTS THE LETTER C

IF WE WERE TO POKE THIS NUMBER INTO MEMORY LOCATION 1Ø48, WE WOULD SEE:

(A) A FLASHING C
(B) A NORMAL C
(C) AN INVERSE C

QUESTION 4

WHAT NUMBER WAS USED IN EARLY COMPUTERS TO PUT A SWITCH IN THE "ON" POSITION?

QUESTION 5

HOW MANY DIFFERENT NUMBERS CAN BE STORED IN ONE MEMORY LOCATION?

QUESTION 6

FIND THE VALUE OF THIS BINARY NUMBER:

ØØ11ØØ11

QUESTION 7

A MEMORY LOCATION WITH THE ADDRESS
18Ø WOULD BE FOUND ON PAGE ....

QUESTION 8

A 4 DIGIT BINARY NUMBER IS
CALLED A ...

QUESTION 9

BINARY IS BASED ON MULTIPLES OF ...
DECIMAL IS BASED ON MULTIPLES OF ...
HEXADECIMAL IS BASED ON MULTIPLES OF ...

ENTER THE MISSING NUMBERS IN ORDER
BEGINNING FROM THE TOP. SEPARATE
ANSWERS WITH COMMAS.

QUESTION 1Ø

WRITE THE NUMBER 13 IN HEXADECIMAL.

QUESTION 11

TO DIRECTLY LOOK AT AND CHANGE COMPUTER
MEMORY IN HEXADECIMAL, YOU MUST FIRST
BE IN ..... MODE.

QUESTION 12

WHAT COMMAND WILL PUT YOU IN MONITOR
MODE ON THE COMPUTER?

QUESTION 13

ASSUME THAT YOU ARE IN MONITOR MODE.
TYPE THE NUMBER 12 INTO MEMORY LOCATION 74.

QUESTION 14

WHICH OF THESE IS A HEXADECIMAL NUMBER?

(A) E4
(B) -11
(C) 3*2
(D) $13
(E) A AND D

QUESTION 15

WHAT IS THE VALUE OF THIS NUMBER
IN DECIMAL?

F2

## Quiz 4

QUESTION 1

```
A$="TELEPHONE      B$="HOMEWORK
          C$="SETTLE

10 X$ = RIGHT$(A$,5)
20 Y$ = LEFT$(B$,4)
30 Z$ = MID$(C$,2,2)
40 ?Z$;" ";X$;" ";Y$
```

WHAT WILL LINE 40 PRINT?

QUESTION 2

```
W$="PRINTED LETTERS"
```

TYPE THE INSTRUCTION THAT WOULD TELL YOU IF W$ CONTAINS ANY HIDDEN CHARACTERS.

QUESTION 3

```
5 A$="THAT IS CORRECT"
10 FOR X = 1 TO 4
20 POKE 767 + X,X
30 NEXT X
40 S = PEEK(770)
50 PRINT LEFT$(A$,S)
```

WHAT WILL LINE 50 PRINT?

QUESTION 4

```
10 A$="BEGIN"
20 FOR X = 2 TO 20 STEP 2
30 A$ = A$ + CHR$(X)
40 NEXT X
50 Z = LEN(A$)
```

WHAT WILL BE THE VALUE OF Z HERE?

QUESTION 5

```
10 A$="LONG-EARED UNRULY PACK ANIMAL"
20 W$ = MID$(A$,1,1)
30 X$ = MID$(A$,6,1)
40 Y$ = MID$(A$,27,1)
50 Z$ = MID$(A$,12,1)
60 M$ = Y$ + Z$ + W$ + X$
70 PRINT M$
```

WHAT WILL LINE 70 PRINT?

QUESTION 6

```
10 A$="LONG-EARED UNRULY PACK ANIMAL"
20 W$ = MID$(A$,1,1)
30 X$ = MID$(A$,6,1)
40 Y$ = MID$(A$,27,1)
50 Z$ = MID$(A$,12,1)
60 M$ = Y$ + Z$ + W$ + X$
70 PRINT M$
```

LINE 60 IS AN EXAMPLE OF .....

QUESTION 7

```
A$="J2S3"     B$="3D0G"     C$="66"

Z = VAL(A$) + VAL(B$) + VAL(C$)
```

WHAT IS THE VALUE OF Z HERE?

QUESTION 8

.... WILL CONVERT A NUMBER TO A STRING.

QUESTION 9

STR IS TO NUMBER AS ... IS TO STRING.

QUESTION 10

```
10 FOR R=1 TO 5
20 READ A$
30 C$(R) = A$
40 C(R) = VAL(A$)
50 T = T + C(R)
60 NEXT R
70 PRINT T
80 DATA A PARTRIDGE IN A PEAR TREE,2 TURTLE DOVES,
   3 FRENCH HENS,4 CALLING BIRDS, 5 GOLDEN RINGS
```

WHAT WORD OR WORDS WILL BE STORED
IN C$(3)?

QUESTION 11

```
10 FOR R=1 TO 5
20 READ A$
30 C$(R) = A$
40 C(R) = VAL(A$)
50 T = T + C(R)
60 NEXT R
70 PRINT T
80 DATA A PARTRIDGE IN A PEAR TREE,2 TURTLE DOVES,
   3 FRENCH HENS,4 CALLING BIRDS, 5 GOLDEN RINGS
```

WHAT IS THE VALUE OF T IN LINE 70?

QUESTION 12

```
10 FOR R=1 TO 5
20 READ A$
30 C$(R) = A$
40 C(R) = VAL(A$)
50 T = T + C(R)
60 NEXT R
70 PRINT T
80 DATA A PARTRIDGE IN A PEAR TREE,2 TURTLE DOVES,
   3 FRENCH HENS,4 CALLING BIRDS, 5 GOLDEN RINGS
```

WHAT IS STORED IN C(1)?

QUESTION 13

```
10 FOR R=1 TO 5
20 READ A$
30 C$(R) = A$
40 C(R) = VAL(A$)
50 T = T + C(R)
60 NEXT R
70 PRINT T
80 DATA A PARTRIDGE IN A PEAR TREE,2 TURTLE DOVES,
   3 FRENCH HENS,4 CALLING BIRDS, 5 GOLDEN RINGS
```

WHAT IS STORED IN C(3)

## Quiz 5

QUESTION 1

WHAT COMMAND WOULD YOU USE TO SET
UP THE SCREEN FOR HI-RES GRAPHICS?

QUESTION 2

```
10 HGR
------------
30 HPLOT 15,15 TO 15,35
40 END
```

ADD LINE 20 TO PRINT A GREEN LINE
ON THE HI-RES SCREEN.

QUESTION 3

(T OR F)

VERTICAL POSITIONS ON THE HI-RES
SCREEN ARE NUMBERED FROM 0 TO 279.

QUESTION 4

(T OR F)

```
5 HGR
15 HCOLOR=2
10 HPLOT 5,20 TO 40,20 TO 40,40 TO 5,40
```

THIS PROGRAM WILL DRAW A BOX

QUESTION 5

(T OR F)

```
10 HGR
20 HCOLOR=1
30 HPLOT 12,10 TO 12,100
```

A LINE IS VISIBLE ON THE SCREEN WHEN
THIS PROGRAM IS RUN.

QUESTION 6

```
1Ø HGR
15 FOR X=Ø TO 279
2Ø HCOLOR=6
3Ø HPLOT X,4Ø
4Ø HCOLOR=4
5Ø HPLOT X,4Ø
6Ø NEXT X
```

WHAT WILL THIS PROGRAM PRINT?

(A) A DOTTED BLUE LINE
(B) A DOT MOVING RAPIDLY ACROSS
    THE SCREEN.
(C) NOTHING

QUESTION 7

(T OR F)

IF YOU DO NOT ENTER AN HCOLOR COMMAND IN HGR, ANYTHING PLOTTED WILL BE GRAYISH-WHITE IN COLOR.

QUESTION 8

----- SCREEN MEMORY STARTS AT 16384 AND ENDS AT 24576.

(A) HGR
(B) HGR2
(C) TEXT

QUESTION 9

(T OR F)

```
1Ø HGR
2Ø POKE -16302,Ø
3Ø HCOLOR=6
4Ø HPLOT 1Ø,Ø TO 1Ø,159
```

THIS PROGRAM WILL GIVE YOU A SCREEN FULL OF GRAPHICS.

QUESTION 10

```
10 HGR
20 HCOLOR=2
30 HPLOT 2,2 TO 100,100
40 HOME
50 PRINT"THIS IS A DIAGONAL LINE."
60 END
```

WHEN THIS PROGRAM IS RUN,
WHAT WILL BE ON THE SCREEN?

(A) A LINE
(B) TEXT
(C) A LINE AND TEXT

QUESTION 11

```
10 HGR
20 HCOLOR=1
30 HPLOT 3,3
--------
50 END
```

TYPE A COMMAND ON LINE 40 THAT WILL
CHANGE THE COLOR OF THE SCREEN TO
GREEN.

QUESTION 12

WHICH OF THESE COMMANDS CAN ERASE A
HI-RES SCREEN TO BLACK?

(A) HGR
(B) CALL 62450
(C) CALL 62454
(D) ALL OF THESE

QUESTION 13

TYPE IN THE INSTRUCTION THAT WILL
ALLOW YOU TO DRAW ON HGR2 NO MATTER
WHAT IS BEING DISPLAYED.

QUESTION 14

WHICH INSTRUCTION WILL ALLOW YOU
TO MOVE FROM TEXT BACK TO A PREVIOUS
HGR SCREEN WITHOUT ERASING THAT
SCREEN?

QUESTION 15

(T OR F)

```
10 TEXT
15 REM:SWITCHING SCREENS
20 POKE -16304,0
30 POKE -16299,0
40 POKE -16304,0
```

THIS ROUTINE WILL GO FROM TEXT TO
HGR TO HGR2 AND BACK TO HGR

QUESTION 16

TYPE IN THE INSTRUCTION THAT TELLS
THE COMPUTER TO BEGIN PROGRAM STORAGE
AT ADDRESS 24576

## ANSWERS TO QUIZZES

### Quiz 1

*1.* A; *2.* C; *3.* NUMBER; *4.* PRINT PEEK (50); *5.* POKE 33.15; *6.* 2; *7.* CALL 5; *8.* POKE 32, 8; *9.* 20; *10.* 12; *11.* TEXT; *12.* PRINT PEEK (12000); *13.* F; *14.* C; *15.* B; *16.* 40

### Quiz 2

*1.* PRINT CHR$ (66); *2.* C; *3.* C; *4.* T; *5.* 50. PRINT A$, ASC(A$); *6.* C; *7.* HIMEM: 31000; *8.* F; *9.* A

### Quiz 3

*1.* B; *2.* A; *3.* C; *4.* 1; *5.* 256; *6.* 51; *7.* 0; *8.* NYBBLE; *9.* 2, 10, 16; *10.* D; *11.* MONITOR; *12.* CALL-151; *13.* 74:12; *14.* E; *15.* 242

### Quiz 4

*1.* ET PHONE HOME; *2.* PRINT LEN (W$); *3.* THA; *4.* 15; *5.* MULE; *6.* CONCATINATION; *7.* 69; *8.* STR; *9.* VAL; *10.* 3 FRENCH HENS; *11.* 14; *12.* 0; *13.* 3

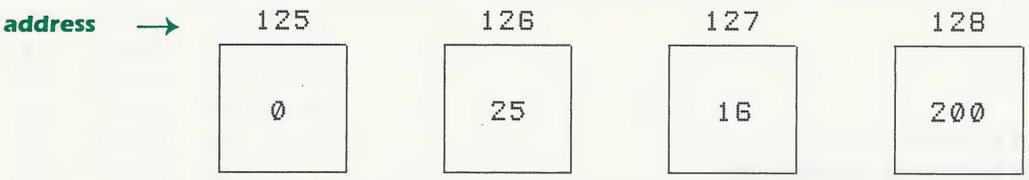
### Quiz 5

*1.* HGR; *2.* 20 HCOLOR = 1; *3.* F; *4.* F; *5.* F; *6.* B; *7.* F; *8.* B; *9.* T; *10.* A; *11.* 40 CALL 62454; *12.* D; *13.* POKE 230, 64; *14.* POKE -16304, 0; *15.* F; *16.* POKE 104, 96

## Lesson 1, Part 1
SUMMARY

**— PEEK —**
**— POKE —**
**— CALL —**

The APPLE II's memory is divided up into individual memory boxes. Each box has an address. A number is stored in each box. This number can represent a number, a letter, or a machine instruction.




**number stored in memory box**

K stands for 1000. 48K of memory stands for approximately 48,000 memory boxes (calculated in binary terms).

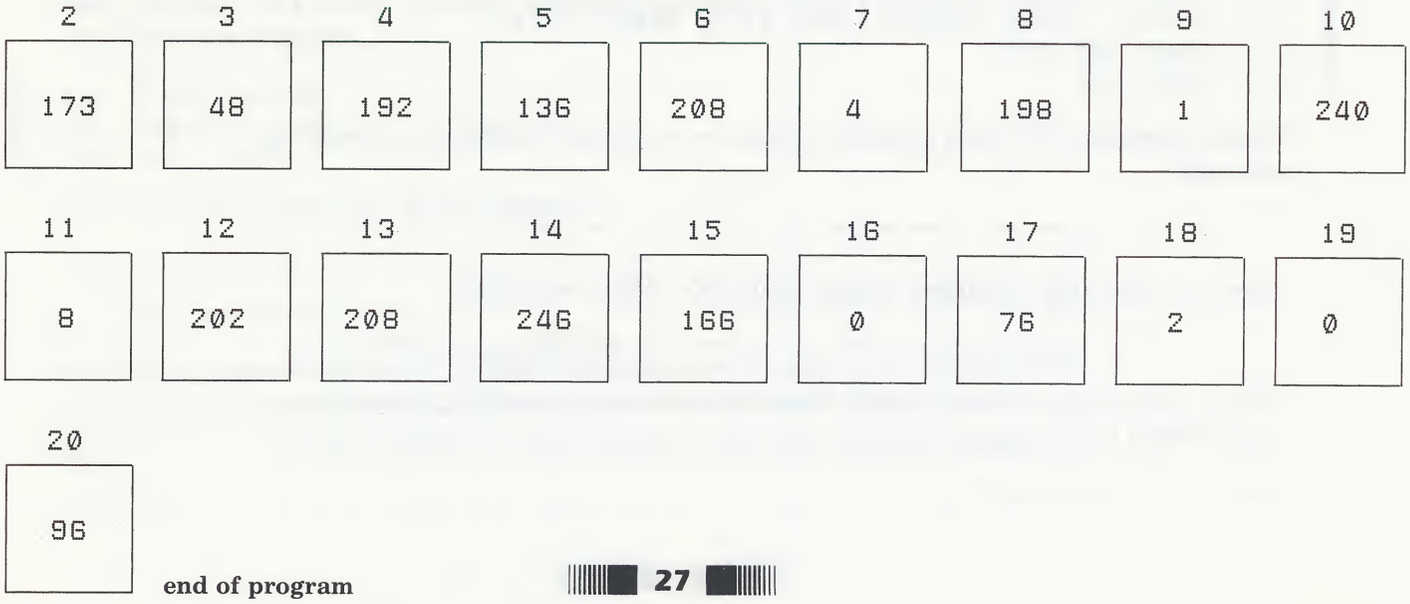
The BASIC programmer can look at the number stored in each box with a **PEEK** instruction. PRINT **PEEK**(33) prints the number stored in memory location 33.

The number in a memory box can be changed with a **POKE** instruction. **POKE** 33,15 stores the number 15 in memory location 33.

The **POKE** instruction should be used very carefully. A wrong **POKE** can interfere with the functioning of the computer and can cause the system to crash.

Machine language instructions are stored in computer memory boxes as numbers. Each instruction is stored in a separate address. Several of these numbers together make up a machine language program.

### MACHINE LANGUAGE PROGRAM




**end of program**

To execute the machine language program illustrated here, when in the BASIC mode, use the **CALL** instruction. **CALL** 2 will activate the above machine language program starting at memory location 2.

## Lesson 1, Part 1
## PRACTICE PROBLEMS

This exercise will use memory location 50. Start by typing in the following direct command:

```
PRINT PEEK (50)
```

What number is printed? _____
Now type in the following commands:

```
POKE 50, 127
PRINT ''SAMPLE TEXT''
```

What did the **POKE** command do to the text printed on the screen?

_____

_____

Now repeat the above commands, but **POKE** 63 into location 50. How does this affect what is printed on the screen?

_____

Now enter and **RUN** this program; limit your numbers to those between 1 and 255.

```
5  HOME
10  PRINT "TYPE IN A NUMBER
12  INPUT X
20  POKE 50,X
30  PRINT "TEXT LOOKS LIKE THIS WHEN X=";X
35  POKE 50,255
40  GOTO 10
```

What happens to the text printed on the screen as each different number is entered?

_____

Now try entering a number larger than 255. What happens?

_____

What is the largest number that X can be before you receive an error message? What is the largest number that can be poked into a memory box?

_____

Enter and **RUN** the following program, which puts a machine-language program into memory through **POKE** instructions.

```
10  POKE 768,173
20  POKE 769,48
30  POKE 770,192
40  POKE 771,136
50  POKE 772,208
60  POKE 773,4
70  POKE 774,198
80  POKE 775,7
90  POKE 776,240
100  POKE 777,8
110  POKE 778,202
120  POKE 779,208
130  POKE 780,246
140  POKE 781,166
150  POKE 782,6
160  POKE 783,76
170  POKE 784,0
180  POKE 785,3
190  POKE 786,96
```

Type **RUN** and press RETURN. The machine-language instructions will be entered, but nothing that you can observe on the monitor will happen. However, the new numbers are now resident in memory.

Now type **NEW** and enter the following program:

```
10  FOR M = 768 TO 786
20  PRINT "MEMORY BOX ";M;" CONTAINS ";
30  PRINT  PEEK (M)
40  NEXT M
```

**RUN** the above program. Are the machine-language instructions still in the memory boxes as you **POKE**d them in?

_____

Now you can **CALL** the machine-language program. Type NEW and enter the following program.

```
10  POKE 6,100
20  POKE 7,100
30  CALL 768
```

Run the above program. What happens?

_____

_____

Now change the values being **POKE**d into memory boxes 6 and 7 and **RUN** the program again. Can you figure out what box 6 does? What 7 does?

BOX 6: _____

BOX 7: _____

The following program will generate music. You could insert this routine into your own program. Enter and **RUN** the program.

```
10  REM  MUSIC PROGRAM
20  REM  POKE ROUTINE NEED ONLY RUN ONCE
25  POKE 768,173: POKE 769,48: POKE 770,192: POKE 771,136
26  POKE 772,208: POKE 773,4: POKE 774,198: POKE 775,7
27  POKE 776,240: POKE 777,8: POKE 778,202: POKE 779,208
30  POKE 780,246: POKE 781,166: POKE 782,6: POKE 783,76
31  POKE 784,0: POKE 785,3: POKE 786,96
1000  REM  ACTUAL MUSIC SUBROUTINE
1010  READ X,Y: IF X =  - 1 THEN  END
1020  POKE 6,X: POKE 7,Y
1030  CALL 768
1040  GOTO 1000
1900  DATA  95,100,127,50,113,50,127,50
2000  DATA  95,100,127,50,113,50,127,50
2005  DATA  143,50,152,50,127,100,0,50,127,50,113,75,85,50
2010  DATA  95,50,101,50,113,100,101,50,95,150
2011  DATA  -1,0
```

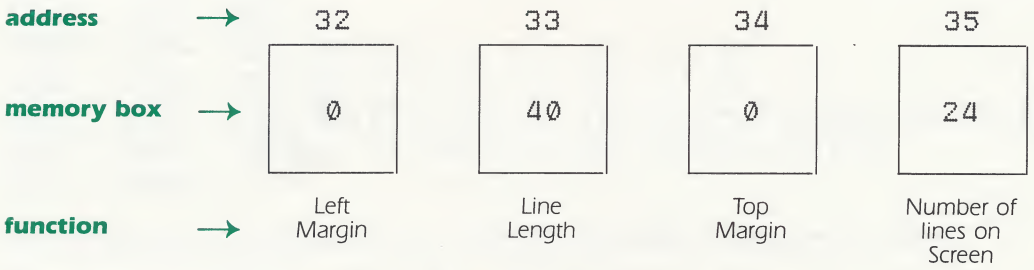
The information entered in lines 1900 through 2011 consists of notes.

Experiment by changing the notes, or the duration of the notes in the data lines of the above program. In each pair of numbers, the first represents the note and the second represents the duration. Exhibit 1 shows APPLE II tones. It gives values for two octaves of APPLE notes. Use these values to change the data lines 1900 through 2011.

## Lesson 1, Part 2
## SUMMARY

— Negative Addresses —
— Positioning Type On Screen —
— APPLE II Speaker —
— Default Values —
— Direct Keyboard Entry —

Memory locations 32, 33, 34 and 35 control how text is positioned on the screen.




Default values are numbers that are automatically put into memory locations when the computer is turned on. The default values for locations 32, 33, 34 and 35 are shown in the boxes above.

The APPLE II can access a memory location with a positive number address or a minus number address. **PEEK**(-151) is the same as **PEEK**(65536-151) or **PEEK**(65385).

Location -16384 is used by the computer for keyboard entry. The location is used by the computer as a port to communicate with the keyboard. Each time a character is typed on the keyboard, it goes to location -16384 first.

Location -16336 affects the APPLE II speaker. Any time memory location -16336 is **PEEK**ed at, a "click" is generated by the speaker.

## Lesson 1, Part 2
## PRACTICE PROBLEMS

Determine what the machine-language CALL routines below do. Enter and **RUN** the following program, which will fill the screen with characters. Then, type in the **CALL** commands and see what happens.

```
10  HOME
20  FOR X = 1 TO 960
30  PRINT "X";
40  NEXT X
50  HTAB 1: VTAB 1: PRINT "YOU ARE HERE"
```

After **RUN**ning the program, enter the following **CALL** commands to see what happens to the material on the screen.

CALL -1998 _____

CALL -912 _____

CALL -936 _____

The following program sets up a text window in the middle of the screen. Enter and **RUN** the program.

```
10  HOME
20  PRINT
30  PRINT "ONLY THE TEXT WINDOW WILL SHOW CHANGE
40  FOR X = 5 TO 23: VTAB X: FOR Y = 0 TO 39
45  PRINT ".";: NEXT Y
50  NEXT X
60  HTAB 15: VTAB 17
70  PRINT "TEXT WINDOW"
80  POKE 32,15: REM  LEFT MARGIN
90  POKE 33,10: REM  LINE LENGTH
100  POKE 34,5: REM  TOP MARGIN
110  POKE 35,15: REM  NUMBER OF LINES
115  HOME
120  SPEED= 200: REM  SPEED THAT TYPE IS PRINTED(255 IS NORMAL)
125  FOR X = 1 TO 20: PRINT "THIS IS ALL THAT YOU WILL SEE."
130  NEXT X
140  SPEED= 255
```

After **RUN**ning the above program, type **HOME.** What part of the screen is cleared? _____

Then type **TEXT.** Where does the cursor go? _____

The following program uses memory location -16384. The program loops and looks at location -16384 to see if a key has been pressed. Each loop is counted until a change appears at the location.

```
10  HOME
20  REM  PROGRAM TO MEASURE SPEED OF RESPONSE
30  VTAB 10: PRINT "PRESS A KEY"
35 K =  PEEK ( - 16384)
37  REM  CHECKS TO SEE IF KEY IS DEPRESSED
40  IF K > 127 THEN  GOTO 100
50 W = W + 1: GOTO 35
100  POKE  - 16368,0: REM  CLEARS OUT KEYBOARD ENTRY
110  VTAB 20: PRINT "TIME TAKEN: ";W;" LOOPS"
120 W = 0
130  PRINT "TO TRY AGAIN, PRESS RETURN";: INPUT A$
140  GOTO 10
```

Replace line 100 with REM. What happens?

_____

_____

Enter and **RUN** the following three programs to see what kind of sound each one generates.

```
10  FOR X = 1 TO 100
20 Z =  PEEK ( - 16336)
30  NEXT
```

Sound generated: _____

```
10 A =  - 16336
20  FOR X = 1 TO 300
30 Z =  PEEK (A)
40  NEXT
```

Sound generated: _____

```
10 T = 75
20  FOR X = 1 TO 100
30 Z =  PEEK ( - 16336)
40  FOR A = 1 TO T
50  NEXT
60 T = T - 1
70  NEXT
```

Sound generated: _____

## Lesson 1, Part 2
NOTES

**RUN QUIZ 1.**

## Lesson 2, Part 1
## SUMMARY

— ASCII Codes —
— Control Characters —
— **CHR$** —
— **ASC** —

The APPLE II stores letters and other characters in the form of numbers. The numerical code used to do this is called American Standard Code for Information Interchange, or ASCII. Exhibit 2 shows the codes used in the APPLE II.

The APPLE II often uses "invisible" characters to signal that certain operations should take place. These invisible characters are called CONTROL CHARACTERS. Control characters can also be typed from the keyboard by holding the CONTROL key down while typing another letter. For example, CONTROL–C (ASCII 3) is made by holding down the control key and typing the letter C. This character can be used to stop an APPLE program.

**CHR$** can be used with an ASCII number to produce a character. **PRINT CHR$**(65) will print the letter A. **PRINT CHR$**(3) is the same as CONTROL–C.

**ASC** is the opposite of **CHR$**. **ASC**("A") produces the ASCII value of the letter A, or 65. **PRINT ASC**(W$) will print the ASCII value of the first letter stored in W$.

## Lesson 2, Part 1
## PRACTICE PROBLEMS

The following program will print all of the characters that the APPLE II will normally print. Enter and **RUN** this program.

```
10  HOME : VTAB 5
20  FOR N = 0 TO 95
30  PRINT  CHR$ (N);
40  NEXT N
```

The following program shows one practical use of the **CHR$** instruction. Enter and **RUN** this program.

```
10  HOME
20  VTAB 3
30  PRINT "THIS IS NOT SHOWN IN QUOTES."
40  VTAB 6
50  PRINT  CHR$ (34);"THIS IS SHOWN IN QUOTES!"; CHR$ (34)
```

As you remember, you cannot use quotation marks in a **PRINT** statement. The **CHR$** instruction allows you to make up a **PRINT** statement that will show quotation marks.

The following program will tell you the ASCII numbers for characters typed from the keyboard. Try ALL keys, including CONTROL, ESC and RETURN.

```
10  HOME
20  VTAB 3
30  PRINT "TYPE IN ANY KEY:"
40  GET C$
45  REM  *GET* IS AN INSTRUCTION THAT TAKES JUST 1
50  REM  CHARACTER FROM THE KEYBOARD
60  PRINT "YOUR CHARACTER HAS AN ASCII VALUE OF ";
70  PRINT  ASC (C$)
80  PRINT : GOTO 30
```

Did you notice that you could not break out of the program? You can break this program *only* by pressing RESET. What line could you enter that would allow you to end the program? _____

Enter and **RUN** the following program:

```
10  HOME
15  VTAB 3
20  PRINT "TYPE IN A WORD:"
30  INPUT W$
40 N =  ASC (W$)
50  PRINT "THE FIRST LETTER IS: "; CHR$ (N)
60  PRINT "AND IT HAS AN ASCII VALUE OF ";N
```

Determine the uses for the following control characters. You may refer to Exhibit 3 at the back of this book.

CONTROL–C _____

CONTROL–X _____

CONTROL–G _____

CONTROL–H _____

CONTROL–M _____

## Lesson 2, Part 2
## SUMMARY

— Memory Map —
— ROM/RAM —
— Screen Memory —

The APPLE II has a memory map. That is, certain areas of memory are used to do specific things. Exhibit 4 shows the APPLE II memory map.

The high end of the APPLE memory locations are usually "**R**ead **O**nly **M**emory," or ROM. The memory locations in ROM cannot be changed using program instructions or by turning off the computer.

The rest of memory is called "**R**andom **A**ccess **M**emory," or RAM, which can be changed by programming or by turning off the computer. A 48K APPLE has about 48000 memory locations in RAM.

The monitor program is stored in ROM. The APPLE II gets its instructions on what to do during power-up from the monitor program. Applesoft BASIC is also stored in ROM on most APPLE IIs.

Applesoft programs are stored in the free memory space starting with memory location 2048 and ending with memory location 38399. As programs are entered, the number of free memory locations decreases. **PRINT FRE**(0) will tell you how much free memory space is available. **HIMEM** can be used to move the top boundary of a BASIC program. **HIMEM:**28000 moves the top boundary down from 38399 to memory location 28000.

Memory locations 1024 to 2047 store information that will appear on the APPLE TV screen. ASCII numbers stored in each location will put a character on the screen. ASCII numbers will appear on the screen as flashing. By adding 128 to the numbers, the characters will appear as normal. By subtracting 64 from the ASCII numbers, the characters will come out as inverse.

## Lesson 2, Part 2
## PRACTICE PROBLEMS

Enter the following command:

```
PRINT PEEK(65000)
```

What number is stored in memory location 65000? _____

Now enter POKE 65000,0 and repeat the above procedure. Did the contents of 65000 change? Why or why not?

_____

Enter and **RUN** the following program:

```
10  HOME : VTAB 3
20  FOR M = 43380 TO 43581: REM  DOS MEMORY LOCATIONS
30  LET NO =  PEEK (M)
40  PRINT  CHR$ (NO);: NEXT M
```

Does the print look familiar? Describe it. _____

The following commands will change the number of memory locations available for BASIC programs.

Enter:

```
HIMEM:30100
PRINT FRE(0)
```

How much memory is now available for BASIC programs?

_____

Now let's return the memory space to normal.

Enter:

```
HIMEM:38400
PRINT FRE(0)
```

Now try this. Enter:

```
HIMEM: 65000
PRINT FRE(0)
```

Did you get a negative number? Because of the old limit of Integer BASIC, **FRE** (0) cannot produce a number larger than 32767. If the number of free memory spaces is larger than this, Applesoft BASIC gives you a negative number. By subtracting the number from (or adding the negative number to) 65536, you will get the correct number.

Try this:

```
PRINT FRE(0) + 65536
```

Now return HIMEM to 38400

Enter and **RUN** this program:

```
10  HOME
20  FOR NO = 0 TO 255
30  FOR M = 1448 TO 1487
40  POKE M,NO
50  VTAB 5
60  PRINT "WE WILL NOW POKE ";NO
70  PRINT " INTO MEMORY BOX #";M
80  NEXT M
90  NEXT NO
```

Now enter and **RUN** this program:

```
10  HOME
20  VTAB 10: HTAB 10
30  PRINT "MEMORY LOCATION 1468"
40  PRINT
45  VTAB 2
50  PRINT " TYPE IN A NUMBER ";
60  INPUT NO
65  CALL  - 958: REM  CLEARS PART OF SCREEN
69  PRINT
70  PRINT "WE NOW POKE ";NO
75  PRINT "INTO MEMORY LOCATION 1468."
80  POKE 1468,NO
90  GOTO 20
```

What numbers are stored in the screen memory locations when the screen is "blank"?

---

What numbers are stored in the screen memory locations after entering **GR** and then entering **TEXT**?

---

**RUN QUIZ 2.**

## Lesson 3, Part 1
## SUMMARY

— Binary Numbers —
— BYTES —
— BITS —

A computer at its simplest level works by turning switches on and off. An "off" switch is represented by a numeral 0 and an "on" switch by a 1.

Binary numbers are formed by combining two numerals: 0 and 1. Each numeral is put into a "column." Each column is a multiple of 2 working from right to left.

| 2×2×2×2×2 | 2×2×2×2 | 2×2×2 | 2×2 | 2 | 1 |
|---|---|---|---|---|---|
| 32 | 16 | 8 | 4 | | |
| | | | | | |

The value of a binary number is calculated by multiplying the numeral times the value of the column:

| 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |

```
64 × 0 =  0
32 × 1 = 32
16 × 0 =  0
 8 × 0 =  0
 4 × 1 =  4
 2 × 1 =  2
 1 × 0 =  0
-----------
         38
```

0100110 = 38

Each binary numeral is called a bit, which stands for **BI**nary digi**T**:

0 1 0 0 1 1 0 ← bit

Eight bits make one byte. A byte is a binary number with 8 columns, each filled with either an 0 or a 1. For example:

```
1 0 0 0 0 0 0 1 ←
0 0 0 0 0 1 1 1 ← BYTE
```

Bytes are shown with 8 bits, or columns, even if some of the columns must be filled with Øs.

The largest possible byte is represented:

1 1 1 1 1 1 1 1 = 255

The smallest possible byte is represented:

Ø Ø Ø Ø Ø Ø Ø Ø = Ø

## Lesson 3, Part 1
## PRACTICE PROBLEMS

The columns used in binary numbers are multiples of 2. Some of these values will be important when you write in BASIC. To get started, fill in the values of each column in a binary byte below. Do your own math by multiplying each successive column by 2. (Start at the far right.)

Using the above columns, figure out the values of the following bytes. The answers appear upside-down at the bottom of the page.

1 1 Ø Ø Ø Ø Ø 1 _____

Ø Ø 1 1 1 1 1 Ø _____

1 Ø 1 1 1 1 1 1 _____

Ø 1 1 1 1 1 1 Ø _____

Ø Ø Ø Ø Ø Ø 1 1 _____



Answers to Problems

193
62
191
126
3

A "flag" is a switch in the computer that means something when it is turned on or off. A single bit inside a byte often serves as a flag. The following example is designed to demonstrate to you just what a flag is.

A programmer who is working in binary numbers can spot these flags easily. For example, let's say that the 64 bit is being used as a signal to the computer. Each of the following numbers has the 64 bit "on," so that this particular signal is given when each of these numbers is used:

```
Ø  1  Ø  Ø  Ø  Ø  Ø  Ø = 64
1  1  Ø  1  Ø  Ø  Ø  1 = 209
1  1  1  1  1  1  1  1 = 255
```

A programmer working in binary numbers can quickly spot whether the 64 bit is "on" or "off." The programmer working in BASIC, on the other hand, cannot see this. Is the 64 bit turned "on" or "off" in the number 180? To answer this question, you have to convert the number to binary.

The following will show you an example of how these binary flags work. Let's go back to the computer. Enter and **RUN** the following program. (This program was first introduced in Lesson 2.)

```
10  HOME
20  VTAB 10: HTAB 10
30  PRINT "MEMORY LOCATION 1468"
40  PRINT
45  VTAB 2
50  PRINT " TYPE IN A NUMBER ";
60  INPUT NO
65  CALL  - 958: REM  CLEARS PART OF SCREEN
69  PRINT
70  PRINT "WE NOW POKE ";NO
75  PRINT "INTO MEMORY LOCATION 1468."
80  POKE 1468,NO
90  GOTO 20
```

When you ran the above program, did you remember that ASCII values came out on the screen as flashing? —that ASCII values minus 64 came out as inverse? —that ASCII plus 128 came out as normal?

Now try the following experiment. Enter the following values when the above program is run:

| byte | | | | | | | | value | letter |
|---|---|---|---|---|---|---|---|---|---|
| Ø | Ø | Ø | Ø | Ø | Ø | Ø | 1 | 1 | A |
| Ø | Ø | Ø | Ø | Ø | Ø | 1 | Ø | 2 | B |
| Ø | Ø | Ø | Ø | Ø | 1 | 1 | 1 | 7 | G |
| Ø | Ø | Ø | 1 | 1 | Ø | 1 | Ø | 26 | Z |

Now, go through each number above and turn the bit in the 64 column to a 1. Figure out the value and enter these values in the above program. Does the letter change? What does change?

Now go through and change the bit in the 128 column to a 1. Enter each number again. How do the results change?

---

The bit in the 64 column and the bit in the 128 column represent "signals" to the computer. When the 128 bit is "up" (a 1) the APPLE is told to make the character shown on the screen normal. If it is "down" (Ø) the characters are not normal. The bit in the 64 column tells the computer whether or not to flash the letter.

Bits that act as signals are called "flags." When working on the APPLE, look for the "magic" numbers of 64 and 128. They have special significance.

The **NORMAL, INVERSE** and **FLASH** instructions take ASCII numbers going to the screen memory area and automatically alter them to make it normal, inverse or flashing. The **NORMAL** instruction (which is also the default condition) automatically sets the 128 bit to 1. When the **INVERSE** instruction is given, all ASCII numbers have the 64 bit changed to Ø before going to the screen memory.

## Lesson 3, Part 1
NOTES

## — Hexadecimal Numbers —
## — Machine-Language Monitor —
## — NYBBLE —

The APPLE II uses binary numbers at its most basic level. Each memory address consists of a 16 bit number (2 bytes). The number stored at each address is 8 bits, or 1 byte, long.

The APPLE II memory map is divided up into groups of 256 address locations. Each of these 256 locations makes up 1 page of memory.

Binary numbers are too long to print on the computer screen. In order to shorten the printing of these numbers, the APPLE II uses hexadecimal numbers, called hex for short. Hexadecimal numbers are based on units of 16. Hex is used in this case because binary numbers can be quickly converted to hex and back again by the computer.

To make a hex number, the binary number is first divided up into 4 bit groupings called nybbles. A nybble can equal any decimal number from 0 to 15. Each nybble is then represented by a character that ranges in value from 0 to 15. Each character is then placed in a column that has a value equal to a multiple of 16.

Here are the characters used to show hex numbers:

| DECIMAL | HEXADECIMAL | BINARY |
|---|---|---|
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |
| 16 | 10 | 10000 |

Here is an example of a conversion of binary to hex:

```
  11100011
| 1110 | 0011 |
| (14) |  (3) |
|   E  |   3  |
```

Here is how the hex number is then converted to decimal:

| 16 | 1 |
|---|---|
| E | 3 |

```
(16 X 14) + (3 X 1) = 224 + 3 = 227
```

The APPLE II has a monitor mode in which the user can directly look at and change the computer memory in hex.

**CALL** -151 puts the APPLE II in the monitor mode. An * will appear as a prompt instead of the Applesoft ].

Typing in the address (in hex) and pressing return will show the user what is stored at a particular address.

Typing in the address (in hex) and pressing RETURN will show the user what is stored at a particular address.

To get out of the monitor mode:

1. If you have an APPLE II+:
   just press RESET.
2. If you have an old APPLE II:
   enter 3D0G and press RETURN.

## Lesson 3, Part 2
## *PRACTICE PROBLEMS*

In this practice session you will be working with hexadecimal numbers. You will be able to see a direct application of how the nybble is used by the APPLE II.

Enter a **GR** command and press RETURN. This will put the top part of the screen in the graphics mode.

Next, enter the monitor mode by entering a **CALL** -151.

Now enter the following:

```
704:01
705:12
706:55
707:AF
708:F1
```

The hexadecimal memory addresses given above are part of screen memory. When you enter numbers into this memory, however, you get color squares instead of letters because you are in the graphics mode.

You will notice that as each location is turned on, two different colors appear. There are two nybbles in each memory location. Each nybble determines the color of a square in the memory location.

If you are using a color monitor or TV, determine the color produced by each nybble by typing in different values into the memory locations above.

| NYBBLE VALUE | COLOR |
|---|---|
| 0 | _____ |
| 1 | _____ |
| 2 | _____ |
| 3 | _____ |
| 4 | _____ |
| 5 | _____ |
| 6 | _____ |
| 7 | _____ |
| 8 | _____ |
| 9 | _____ |
| A | _____ |
| B | _____ |
| C | _____ |
| D | _____ |
| E | _____ |
| F | _____ |

Now enter this instruction: C051:00

This puts the computer in the all text mode. Each pair of color squares gives us one letter.

Now enter different hex values into the screen memory locations and see what happens on the screen.

Look at this instruction:

```
704:A1
```

Write a BASIC **POKE** instruction that will do exactly the same thing as this monitor mode instruction. (HINT: in order to do this, first convert the above hex numbers into decimal.)

Now let's get out of the monitor mode. On the APPLE II+ simply press RESET. On older APPLE IIs, type in 3D0G and press RETURN. This should return the Applesoft prompt (]) to the screen.

## Lesson 3, Part 2
NOTES

**RUN QUIZ 3.**

— String Logic —
— LEN —
— A$ —
— LEFT$ —
— RIGHT$ —
— MID$ —

String variables can store any ASCII character, including control characters. The **LEN** function will return the number of characters stored in a string.

Example:

```
A$ = ''LETTERS''
PRINT LEN(A$)
7
```

**LEN** will also count invisible characters.

**LEFT$, MID$,** and **RIGHT$** are functions used to separate out pieces of a string. **LEFT$** separates characters to the left.

Example:

```
A$ = ''1234567890''
PRINT LEFT$(A$,6)
123456
```

**RIGHT$** separates characters to the right.

Example:

```
A$ = ''1234567890''
PRINT RIGHT$(A$,6)
567890
```

**MID$** starts at a point within the string and separates out a specified number of characters. The first number in the instruction tells the computer where to start, and the second number tells how many characters to separate out.

Example:

```
A$ = ''1234567890''
PRINT MID$(3,4)
3456
```

## Lesson 4, Part 1
## *PRACTICE PROBLEMS*

Enter and RUN the following program. Type various characters into A$, including "invisible" ones (for example, CONTROL-G).

```
10  HOME
20  VTAB 3
30  PRINT "TYPE CHARACTERS INTO A$:"
35  INPUT A$
40  PRINT "A$ CONTAINS ";
50  PRINT  LEN (A$);" CHARACTERS."
60  PRINT : GOTO 30
```

Note that line 50 causes the computer to count characters, even those that do not appear on the monitor.

**LEN** can also be used in a **FOR-NEXT** loop. Enter and **RUN** this program:

```
10  HOME
20  VTAB 3
30  PRINT "TYPE CHARACTERS INTO A$:"
40  INPUT A$
50  PRINT "A$ CONTAINS"
60  PRINT
70  FOR C = 1 TO  LEN (A$)
80  PRINT C;" CHARACTERS"
90  NEXT C
```

Set A$ equal to some string, such as "12345 . . .". Now separate out the characters using **PRINT LEFT$** (A$,N).

What happens when N is larger than the string itself?

---

Do the same thing with **RIGHT$.**

Enter and **RUN** this program, which will put a pattern on the screen. Try several different names.

```
10  HOME
20  VTAB 3
30  PRINT "TYPE IN YOUR NAME:";
35  INPUT N$
40  PRINT
50  PRINT "NAME PYRAMID"
60  PRINT
70  FOR C = 1 TO  LEN (N$)
80  PRINT  LEFT$ (N$,C)
90  NEXT
```

Now change the above statement in line 80 to **RIGHT$** and **RUN** the program.

**MID$** can be used to separate words out of sentence strings. It can be used to look for individual characters in a string.

Enter and **RUN** the following program, which will separate out your first name.

```
10  HOME
20  VTAB 3
30 C = 1: PRINT "TYPE IN YOUR WHOLE NAME:"
40  INPUT N$
50  PRINT
60  IF  MID$ (N$,C,1) = " " THEN  GOTO 90
70  IF C =  LEN (N$) THEN  GOTO 100
80 C = C + 1: GOTO 60
90  PRINT "YOUR FIRST NAME IS "; LEFT$ (N$,C)
95  END
100  PRINT "DON'T YOU HAVE A LAST NAME?"
110  PRINT : GOTO 30
```

Try to write a program that will determine your last name.

Enter and **RUN** this program, which will rearrange the way your name is printed.

```
5  HOME
10  PRINT "INPUT YOUR NAME WITH A SPACE"
20  PRINT "BETWEEN FIRST NAME AND LAST:": PRINT : PRINT
30  REM
40  REM  SPLIT NAME
50  REM
60  INPUT A$:X = 1
70  IF  MID$ (A$,X,1) <  > " " THEN X = X + 1: GOTO 70
80 B$ =  LEFT$ (A$,X - 1)
90 C$ =  RIGHT$ (A$, LEN (A$) - X)
100  PRINT : PRINT
110  PRINT "IS YOUR FIRST NAME ";B$
120  PRINT "OR IS IT ";C$;"?"
130  PRINT : INPUT D$
200  IF C$ = D$ THEN HO$ = B$:B$ = C$:C$ = HO$
210  IF B$ <  > D$ THEN  GOTO 100
220  PRINT
225  PRINT "IN A PHONE BOOK YOUR NAME"
227  PRINT "WOULD BE LISTED AS:"
230  PRINT C$;", ";B$
240  END
```

Now **RUN** the program making an error—forget the space between your first and last name. What happens? Hint: look at line 70.

One use of **MID$** is to examine sentences to find specific words. The following programs show how this is done.

The following program stores three-letter words in a string variable, and then separates out each word individually.

```
20 A$ = "DOG CAT RAT SAT GET KIN FUN SUN RUN"
30  FOR W = 1 TO 33 STEP 4
40  PRINT  MID$ (A$,W,3)
50  NEXT W
```

This program looks into a sentence to find a specific word.

```
20  PRINT : PRINT "TYPE IN A SENTENCE:"
25  PRINT : PRINT
26  INPUT A$
27  PRINT
30  FOR W = 1 TO  LEN (A$)
35 J$ =  MID$ (A$,W,3)
40  IF J$ = "THE" THEN  PRINT "YOU USED 'THE'": END
50  NEXT W
60  PRINT "YOU DID NOT USE THE WORD 'THE'"
```

The following program analyzes the sentence you type in in order to find words from a list of key words. It then fashions an answer based on finding the key word.

```
5  DATA  BAD,LOUSY,AWFUL,TERRIBLE,NOT GOOD
10  DATA  NOT TOO GOOD,NOT VERY GOOD,NOT WELL
20  TEXT : HOME
25  VTAB 3
26  PRINT "HOW ARE YOU TODAY?"
27  INPUT A$
30  FOR V = 1 TO 8
40  READ B$
50  FOR C = 1 TO  LEN (A$)
60  IF B$ =  MID$ (A$,C, LEN (B$)) THEN  GOTO 100
70  NEXT C
80  NEXT V
90  PRINT : PRINT "I'M GLAD TO HEAR IT.": END
100  PRINT : PRINT "THAT'S TOO BAD.": END
```

Note that it's possible to come up with a negative response that will cause the computer to respond "I'M GLAD . . .". For example: THIS HAS BEEN THE WORST DAY OF MY LIFE."

How could you fix this? _____

_____

_____

## Lesson 4, Part 2
## SUMMARY

— Concatination —
— **VAL** —
— **STR** —
— String Arrays —

Concatination (the joining of the contents of two string variables) is handled on the APPLE II as follows:

```
A$ = ''FIRST''        B$ = ''LAST''
            C$ = A$ + B$
            C$ = ''FIRSTLAST''
```

**VAL** converts a string of number characters to numbers. Example:

```
N$ = ''33''
N = VAL(N$)
PRINT N/2
16.5
```

The **VAL** function will separate out any numbers in the string from other characters. If there are no numbers, **VAL** will return a 0. Example:

```
N$ = ''21 YEARS OLD''
PRINT VAL(N$)
21
N$ = ''TWENTY-ONE YEARS OLD''
PRINT VAL(N$)
0
```

**STR** is the opposite of **VAL.** It converts a number so that it can be stored in a string variable. Example:

```
N = 21
N$ = ''I AM'';STR(N);'' YEARS OLD''
PRINT N$
I AM 21 YEARS OLD
```

A string array is a quick way to set up a table of string variables, each with a similar name. The following program sets up a table of 11 variables named N$:

```
10  DIM N$(10)
20  FOR X = 0 TO 10
30  READ N$(X)
40  NEXT X
50  DATA  ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX
60  DATA  SEVEN,EIGHT,NINE,TEN
```

## Lesson 4, Part 2
## *PRACTICE PROBLEMS*

Set a string variable equal to a set of numbers and characters. Then, print the contents of the variable with the **VAL** function. What is printed?

Example:

```
A$ = ''ABC12DE34''

PRINT VAL(A$)

0
```

Enter the following program and **RUN** it. The program will change a message into code.

```
10  REM  SECRET CODE PROGRAM
20  HOME
30  VTAB 3
40  PRINT "TYPE YOUR MESSAGE, AGENT 002:"
50  INPUT M$
60  REM  LET'S SCRAMBLE
70  FOR X = 1 TO  LEN (M$)
80 L$ =  MID$ (M$,X,1)
90 C =  ASC (L$): REM  CONVERT CHARACTER TO NUMBER
100 C = C + 1
105  REM  CHANGE CHARACTER TO NEXT ONE ON ASCII TABLE
110 LL$ =  CHR$ (C): REM  CHANGE NO. BACK TO CHARACTER
120 SE$ = SE$ + LL$: REM  MERGE CHARACTER INTO STRING
130  NEXT X
140  PRINT
150  PRINT "YOUR SCRAMBLED MESSAGE 002:"
160  PRINT
170  PRINT SE$
```

Which line in the program changes the message into code?

The above code is very simple to break. Change the above program so that messages are not so easy to break. Do this by changing line 100.

Now write a program that can decode the messages produced by your code program. Test the program by writing down some secret messages, and then entering them back into the computer when your decode program is running. Do this by changing one line in the program.

Enter the following program and **RUN** it.

```
10  REM  PROGRAM TO WRITE WORDS BACKWARDS
20  HOME
30  VTAB 3
40  PRINT "TYPE IN YOUR MESSAGE:"
45  PRINT
50  INPUT N$
60  FOR X =  LEN (N$) TO 1 STEP  - 1
70 NN$ =  MID$ (N$,X,1)
80 BA$ = BA$ + NN$
90  NEXT X
100  PRINT
110  PRINT "THE BACKWARD MESSAGE IS:"
115  PRINT
120  PRINT BA$
```

Combine the above program with the code program to get even more complex code. Then write a program that straightens out the backward code. (HINT: you will need to add lines between line 70 and line 80. The answer is upside down at the bottom of this page.

```
4 AR = 0:G = 0
5  DIM W$(20)
6  DIM S$(20)
10  REM  PROGRAM TO PUT WORDS IN ALPHABETICAL ORDER
20  REM  BY FIRST LETTER ONLY
30  HOME
40  VTAB 3: PRINT "TYPE IN A WORD"
45  PRINT "OR PRESS RETURN WITHOUT"
50  PRINT "A WORD TO STOP."
60  INPUT W$(AR)
70  IF W$(AR) = "" THEN  GOTO 100
80 AR = AR + 1: GOTO 60
100  PRINT
105  PRINT "SORTING...": PRINT
110  FOR AS = 65 TO 91
115  REM  PRODUCES ASCII CODE FOR A THRU Z
120  FOR X = 0 TO AR
125 T$ =  LEFT$ (W$(X),1)
130  IF T$ =  CHR$ (AS) THEN S$(G) = W$(X):G = G + 1
140  NEXT X
150  NEXT AS
160  FOR G = 0 TO AR
170  PRINT S$(G)
180  NEXT G
```

To get an unreadable scrambled message, add the following lines to the above program:

```
71 C = ASC (NN$)
72 C = C + 1
73 NN$ = CHR$(C)
```

Now you have a message that is scrambled and backwards. In order to get the message translated back, use the above program with the following change:

```
72 C = C - 1
```

Type in the backward, scrambled message into the changed program, and the message will be automatically straightened out!

Use the following list in the above program: APPLE, ZIPPER, BEAR, MANGO, CAT, KANGAROO, SOAP, X-RAY, PELICAN, FISH.

Now **RUN** the above program using this list: APPLE, COW, CAT, AIM, MELT, MAT, BEAR, BAT

Did the above program correctly alphabetize the second list? Why or why not?

_____

## Lesson 4, Part 2
NOTES

**RUN QUIZ 4.**

## — HGR —
## — HCOLOR —
## — HPLOT —
### — Screen Memory —

The **HGR** statement clears the screen and sets it up for high-resolution (hi-res) graphics.

Text will appear at the bottom four lines of the **HGR** screen. To print there, the **VTAB** instruction should be used to position the cursor.

Eight numbers representing 6 colors are available to color the points on the hi-res screen.

**COLOR CHART**

| COLOR NUMBER | COLOR |
|---|---|
| 0 | Black |
| 1 | Green |
| 2 | Violet |
| 3 | White |
| 4 | Black |
| 5 | Red |
| 6 | Blue |
| 7 | White |

**HCOLOR** chooses the color. Example:

```
HCOLOR=5
```

**HPLOT** creates the point on the screen.

In **HGR** there are:

280 horizontal positions numbered 0 to 279, and 160 vertical positions numbered 0 thru 159.

**HPLOT** 0,80 plots a point at 0 horizontal and 80 vertical.

**HPLOT** 0,80 to 270,0 plots a line.

The hi-res screen is divided up into memory locations, with each point on the screen being a bit. Each memory location consists of 7 bits that appear on the screen horizontally, with the 8th bit being a color bit. Each bit that appears on the screen can be either on or off, or one of two colors. Even-numbered bits can be violet or blue—odd-numbered bits green or red. The particular color in the odd or even column depends on whether the color bit is turned on or off.

**Columns In Hi–RES Graphics**

The color of vertical lines depends on whether the line is in an even or odd numbered column. If **HCOLOR** is an even number, then the vertical line must be drawn in an even-numbered column. If **HCOLOR** is odd, then the vertical line must be drawn in an odd-numbered column, or the line will not show on the screen.

**If HCOLOR = 1, a line drawn in column 25, an odd column, would be seen as green on the screen. A line drawn in column 100, an even column, would not appear on the screen.**

**If HCOLOR = 2, a line drawn in column 25 would not appear, but a line drawn in column 100 would be violet.**

**Figures 1, 2, and 3 are examples of how bits are set up in memory locations. You will notice that the 8th or color bit is dotted. This is done to indicate that this bit is not shown on the screen.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Ø | Ø | Ø | Ø | Ø | Ø | Ø | 1 |

Figure 1

**In Figure 1, the color bit is off, and the color of the screen bit will be violet or green.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | Ø | Ø | Ø | Ø | Ø | Ø | 1 |

Figure 2

**In Figure 2, the color bit is on, and the color of the screen bit will be red or blue.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | Ø | Ø | Ø | Ø | Ø | 1 | 1 |

Figure 3

**In Figure 3, the bits turned on are adjacent to one another, producing a white dot on the screen.**

## Lesson 5, Part 1
## PRACTICE PROBLEMS

Colors on the APPLE II screen are affected by both the color chosen and the position plotted on the screen. Enter and **RUN** these programs to see the result of different combinations of color numbers and screen positions.

```
10  HGR : HOME
20  VTAB 21
30  PRINT "INPUT A ROW NUMBER";: INPUT N
40  PRINT "INPUT A COLOR NUMBER": INPUT C
50  HOME : VTAB 21
60  PRINT "ROW #";N,"COLOR #";C
65  HCOLOR= C
70  HPLOT 5,N TO 279,N
75  PRINT "PRESS RETURN TO CONTINUE";: INPUT A$
80  HOME : CALL 62450: GOTO 20
```

```
10  HGR : HOME
20  VTAB 21
30  PRINT "INPUT A COLUMN NUMBER ";: INPUT N
40  PRINT "INPUT A COLOR NUMBER ";: INPUT C
50  HOME : VTAB 21
60  PRINT "COLUMN #";N,"COLOR #";C
70  HPLOT N,5 TO N,150
75  PRINT "PRESS RETURN TO CONTINUE";: INPUT A$
80  HOME : CALL 62450: GOTO 20
```

The following program will **POKE** numbers directly into screen memory locations for the hi-res screen. The idea here is to **POKE** a specific byte number into the memory location and watch how the size, position and color of the resulting dots appearing on the screen change as the numbers change.

```
10  HGR
20  HOME : VTAB 21: PRINT "ENTER A NUMBER BETWEEN 0 AND 255"
22  INPUT A
25  FOR R = 9000 TO 9040 STEP 8: POKE R,A: NEXT
26  VTAB 23: PRINT "PRESS RETURN TO CONTINUE";: INPUT A$
30  GOTO 20
```

Now convert the numbers that you are putting into the screen memory locations over to binary. How do the dots on the screen correspond to the bits in a binary number?

_____

_____

The following program creates some interesting patterns in hi-res graphics. Enter and **RUN** this program:

```
5  FOR R = 768 TO 786: READ A
7  POKE R,A: NEXT R
10  HGR
20 J =  INT ( RND (1) * 7) + 1
30  HCOLOR= J
40 H =  INT ( RND (1) * 150) + 60
50 V =  INT ( RND (1) * 100) + 50
55  HPLOT H,V
60  FOR Z = 1 TO 6
70  HPLOT H - Z,V + Z TO H + Z,V + Z
72  HPLOT H - Z,V - Z TO H + Z,V - Z
80  POKE 6,J * 10: POKE 7,4: CALL 768
90  NEXT Z: GOTO 20
100  DATA  173,48,192,136,208,4,198,7,240
102  DATA  8,202,208,246,166,6,76,0,3,96
```

After **RUN**ning the above program, change lines 40, 50 and 60 to read:

```
10  HGR
20 J =  INT ( RND (1) * 7) + 1
30  HCOLOR= J
40 H =  INT ( RND (1) * 258) + 10
50 V =  INT ( RND (1) * 150) + 10
55  HPLOT H,V
60  FOR Z = 1 TO 10
70  HPLOT H - Z,V + Z TO H + Z,V + Z
72  HPLOT H - Z,V - Z TO H + Z,V - Z
80  POKE 6,J * 10: POKE 7,4: CALL 768
90  NEXT Z: GOTO 20
```

What happened to the patterns and screen positions? _____

_____

This program uses changes in **HCOLOR** to plot and then erase points on the screen, creating the effect of animation. Enter and **RUN** it.

```
10  HGR : HCOLOR= 6
11  FOR X = 100 TO 150
12  HPLOT 125,X TO 145,X
13  NEXT X
14  HCOLOR= 7: HPLOT 136,60 TO 136,99
20  FOR X = 60 TO 99
22  HCOLOR= 2: HPLOT 136,X
24  FOR W = 1 TO 100: NEXT W
26  HCOLOR= 4: HPLOT 136,X - 1
28  NEXT X
30  FOR R = 1 TO 3
35  HCOLOR= 2: HPLOT 2,2
40  CALL 62454: HCOLOR= 5: HPLOT 0,0
45  CALL 62454: NEXT R
```

The next program shows how **HPLOT** statements and changes in color can be erased to create animation. Enter and **RUN** it.

```
10  HGR
55  HCOLOR= 6:H = 150:V = 80: HPLOT H,V
60  FOR Z = 1 TO 40
70  HPLOT H - Z,V - Z: HPLOT H + Z,V - Z
80  HPLOT H - Z,V + Z: HPLOT H + Z,V + Z
90  NEXT Z: HPLOT H - Z,V - Z TO H + Z,V - Z
95  HPLOT H - Z,V + Z TO H + Z,V + Z
100  HCOLOR= 2: FOR Z = 2 TO 40:
     HPLOT H - Z,V - Z TO H + Z,V - Z: NEXT
140  FOR Z = 40 TO 1 STEP  - 1: HPLOT H,V TO H,V + 39
150  HCOLOR= 4: HPLOT H - Z + 1,V - Z TO H + Z - 1,V - Z
155  HCOLOR= 2: HPLOT H - Z,V + Z TO H + Z,V + Z
160  FOR W = 1 TO 100: NEXT : NEXT
```

## Lesson 5, Part 1
NOTES

## Lesson 5, Part 2
## SUMMARY

— **HGR** —
— **HGR2** —
— Animation —
— Memory Mapping —

There are two high-resolution screens available to APPLE II programmers: **HGR** and **HGR2. HGR** has 4 lines of text at the bottom of the screen in the default mode, while **HGR2** provides a full screen of graphics.

Here are the memory locations taken up by hi-res and BASIC programs:

```
2048 to 8191—BASIC program area

8192 to 16383—hi-res screen 1

16384 to 24575—hi-res screen 2

24576 to 38399—BASIC program area
```

All BASIC programs are automatically loaded beginning at 2048, unless the programmer deliberately changes the loading location. Changing the beginning memory location to 24576 gives the programmer 13K of memory to work with.

To put a program above the hi-res area, enter these two commands before loading the program:

```
POKE 104,96

POKE 24576,0
```

Location 104 is part of the "beginning program pointer" needed by the system to find the program.

The value 96 stands for 9600 in hex, which translates into 24576 in decimal.

Location 24576 is the actual address where program storage should begin.

The value 0 must be contained in the first byte of the storage area.

IMPORTANT **POKE**s AND **CALL**s FOR HI-RES GRAPHICS

**POKE** -16304,0—This will go back to hi-res without clearing the screen. It will show the last picture made in hi-res.

**POKE** -16303,0—This will return to the text screen from hi-res. This works something like the **TEXT** instruction.

**POKE** -16299,0—This will display the second hi-res screen. If the program was originally in screen 1, this **POKE** will show you hi-res screen 2, but **HPLOT**ing will still take place in screen 1.

**POKE** -16300,0—This will display the first hi-res screen. If you were originally in hi-res screen 2 before the **POKE, HPLOT**ing will still continue on screen 2.

**POKE** -16302,0—This will get rid of the 4 text lines at the bottom of either hi-res screen.

**POKE** -16301,0—This will put 4 text lines at the bottom of either hi-res screen (even **HGR2**).

**POKE** 230,32—This will cause all subsequent hi-res work to be done on hi-res screen 1, no matter what is on the screen.

**POKE** 230,64—This will cause all subsequent hi-res work to be done on hi-res screen 2, no matter what is on the screen.

**CALL** 62450—This will erase a hi-res page to black.

**CALL** 62454—This will erase a hi-res page and change its entire color to the last **HCOLOR HPLOT**ted.

## Lesson 5, Part 2
## PRACTICE PROBLEMS

The following program produces animation by flipping back and forth between **HGR** and **HGR2.** Enter and **RUN** the following program:

```
10  HGR : GOSUB 100: GOSUB 200
20  HGR2 : GOSUB 100: GOSUB 300
25  POKE  - 16304,0
30  FOR T = 1 TO 10: POKE  - 16299,0: GOSUB 400
40  POKE  - 16300,0: GOSUB 400: NEXT T
45  END
100  HCOLOR= 2: HPLOT 110,50 TO 114,50 TO 118,54
101  HPLOT 120,62 TO 118,66 TO 112,70 TO 110,72
102  HPLOT 108,70 TO 102,66 TO 100,62 TO 102,54 TO 106,50
103  HCOLOR= 1: HPLOT 107,60: HPLOT 113,60
104  HPLOT 106,64 TO 110,66 TO 114,64
105  HCOLOR= 6: HPLOT 106,60: HPLOT 114,60
107  HPLOT 105,64 TO 110,66 TO 113,64
110  HPLOT 110,72 TO 110,94
115  RETURN
200  HCOLOR= 2: HPLOT 110,94 TO 116,110 TO 120,106
201  HPLOT 110,94 TO 104,110 TO 100,106
202  HCOLOR= 6: FOR X = 106 TO 114 STEP 4
203  HPLOT X,50 TO X,56: NEXT
205  HPLOT 110,78 TO 122,88 TO 124,86
210  HPLOT 110,78 TO 98,88 TO 96,86: RETURN
300  HCOLOR= 2: HPLOT 110,94 TO 130,106 TO 132,104
301  HPLOT 110,94 TO 90,106 TO 88,104
303  HCOLOR= 6: FOR X = 106 TO 114 STEP 4
304  HPLOT X,46 TO X,50: NEXT
305  HPLOT 110,78 TO 130,72 TO 132,70
310  HPLOT 110,78 TO 90,72 TO 88,70: RETURN
400  FOR W = 1 TO 500: NEXT W: RETURN
```

Now change line 400 to read:

```
400 FOR W = 1 TO 100 : NEXT W
```

Try other values in line 400 to increase or decrease the speed of animation.

See if you can create your own version of the figure by changing the colors or the values of the points plotted in the subroutines.

The next program also flips between **HGR** and **HGR2,** except that the figure also moves across the screen. Notice that data statements are used in drawing the figure and that they are repeatedly used and restored as the figure's position on the screen changes.

Enter and **RUN** the following program, but be careful when entering the data.

```
3  HGR : HGR2 : TEXT
5  HOME : VTAB 10: PRINT "TRYING ANIMATION WITH A BUTTERFLY'S WING"
7  FOR W = 1 TO 1000: NEXT W
10  POKE  - 16304,0:X = 10:Y = 50
20 FL = 0
22  POKE 230,64: RESTORE : GOSUB 100
30  POKE  - 16299,0: POKE 230,32: RESTORE :X = X - 10:FL = 1
35  FOR C = 1 TO 3: HCOLOR= 0: GOSUB 200: NEXT
40 X = X + 20
50  RESTORE : GOSUB 100
60  POKE  - 16300,0: POKE 230,64:FL = 0
65  RESTORE :X = X - 10
70  FOR C = 1 TO 3: HCOLOR= 0: GOSUB 200: NEXT
80  RESTORE :X = X + 20: GOSUB 100: IF X < 240 THEN  GOTO 30
82  END
100  FOR C = 1 TO 3: HCOLOR= C: GOSUB 200: NEXT : RETURN
200  READ A,B,S: HPLOT A + X,B + Y
210  FOR Z = 1 TO S: READ E,F
211  IF C <  > 3 THEN  HPLOT  TO E + X,F + Y: GOTO 220
215  IF FL = 1 THEN  HPLOT  TO E + X,F + Y
217  IF FL = 0 THEN  HPLOT  TO E + X,Y - F + 14
220  NEXT : RETURN
490  DATA  2,7,4,17,7,17,6,0,6,2,7
500  DATA  17,7,1,22,7
510  DATA  5,6,11,4,4,4,2,5,0,9,0,10,1,10,3,12
515  DATA  1,14,1,15,2,16,4,14,6
```

Now delete lines 35 and 70 and **RUN** the program again. What did lines 35 and 70 do in the program?

---

Setting **HCOLOR** to 0 and plotting over the figure erases it. Another method of doing this is to erase the screen itself.

Enter these lines and **RUN** the program again.

```
35 CALL 62450
```

```
70 CALL 62450
```

Did you notice any difference in the speed of animation when the original lines 35 and 70 were replaced by these lines?

Enter and **RUN** the following program. Which of the **POKE** instructions in line 35 erases the text?

```
3  HGR
10  HCOLOR= 6: HPLOT 0,0
11  CALL 62454
12  HOME
20  VTAB 21: PRINT "THESE WORDS WILL DISAPPEAR AND REAPPEAR"
25  PRINT "WHEN YOU PRESS RETURN."
30  FOR R = 1 TO 10: VTAB 3: INPUT A$
35  POKE  - 16302,0: INPUT A$
40  POKE  - 16301,0: NEXT
```

Try to enter and **RUN** a large BASIC program without first changing the address into which it is loaded. Follow these steps:

1. Type **POKE** 140,12—this resets the location to its default value.
2. Now **RUN LES5A** (this is a large BASIC program).

What happened when the program was **RUN?** If the screen is blank, and you cannot continue, press RESET to start over. You will have to type **RUN HELLO** to continue.

**RUN QUIZ 5.**

## Supplementary Practice
# INTRODUCTION TO GAME DESIGN

Now that you have learned something about hi-res graphics, music routines and string arrays, you are ready to write game programs on your APPLE II.

Enter and **RUN** the following two programs which will illustrate some techniques that will be useful in game design. The first program uses the music routines you learned about previously to generate a game called *Name That APPLE Tune.*

```
5  FOR X = 768 TO 786: READ A: POKE X,A: NEXT X
6 TN = 10
10  HOME : VTAB 10: PRINT "WELCOME TO NAME THAT APPLE TUNE"
11  PRINT : PRINT "HOW MANY PLAYERS(1 OR 2)";: INPUT P
12 T = 7: REM :NUMBER OF SONGS IN THE PROGRAM
15  DIM A$(T)
16  REM  DIMENSION THE STRING ARRAY IF YOU HAVE MORE THAN 11 SONGS
18  FOR W = 1 TO T: READ NAME$:A$(W) = NAME$: NEXT
20 R =  INT ( RND (1) * 7) + 1
30 C$ = A$(R)
35  PRINT : PRINT "WHEN YOU RECOGNIZE THE SONG,"
36  PRINT : PRINT "PLAYER 1,PRESS KEY A"
37  IF P = 2 THEN  PRINT : PRINT "PLAYER 2, PRESS KEY L"
38  PRINT : PRINT "PRESS RETURN TO BEGIN ";: INPUT R$
40  READ NAME$
41  IF NAME$ <  > C$ THEN  FOR M = 1 TO 21: READ J: NEXT : GOTO 40
50  POKE -16368,0:READ D
51 TN = TN - 1
52  IF TN = 0 THEN  FOR W = 1 TO 100:M =  PEEK ( - 16336):
    NEXT : GOTO 100
53  READ A,B: POKE 6,A: POKE 7,B * D: CALL 768
55 KB =  PEEK ( - 16384): IF KB < 128 THEN  GOTO 51
57 KB = KB - 128
58  IF KB = 65 THEN Y = 1: GOTO 63
59  IF KB = 76 THEN Y = 2: GOTO 63
60  GOTO 51
63  HOME
64  POKE  - 16368,0: VTAB 15: PRINT "PLAYER ";Y: GOSUB 200
65  IF P = 1 OR C = 1 THEN  GOTO 100
68  IF Y = 2 THEN Y = 1: GOTO 70
69  IF Y = 1 THEN Y = 2
70  PRINT : PRINT "PLAYER  ";Y: GOSUB 200
100  GOSUB 280: PRINT : PRINT "THE CORRECT ANSWER WAS ";C$
105  PRINT : PRINT "DO YOU WANT TO PLAY AGAIN (Y OR N)";
107  INPUT ANS$: IF ANS$ = "N" THEN  END
110 TN = 10: RESTORE :C = 0:
111  FOR W = 1 TO 26: READ A$: NEXT W: GOTO 20
200  REM  ANSWER ROUTINE
250  PRINT "WHAT IS YOUR ANSWER";: INPUT NAME$
251  IF NAME$ = C$ THEN  GOTO 255
252  PRINT "NO"
253  FOR W = 1 TO 100:M =  PEEK ( - 16336): NEXT : GOTO 275
```

```
255  PRINT "CORRECT!":P(Y) = P(Y) + TN: FOR W = 1 TO 10
256 J =  INT ( RND (1) * 50) + 50: POKE 6,J: POKE 7,30:
    CALL 768: NEXT :C = 1
275  RETURN
280  HOME : VTAB 5: PRINT "PLAYER 1": PRINT : PRINT
     "SCORE = ";P(1): IF P = 2 THEN  VTAB 5: HTAB 20: PRINT "PLAYER 2":
     VTAB 7: HTAB 20: PRINT "SCORE =";P(2)
300  RETURN
3000  DATA   173,48,192,136,208,4,198,7,240,8,202,208,246,166,
      6,76,0,3,96
4000  DATA    JOY TO THE WORLD,AULD LANG SYNE,TWINKLE TWINKLE
      LITTLE STAR,HOME ON THE RANGE,BAA BAA
      BLACK SHEEP,SAILING,LONDON BRIDGE
5000  DATA  JOY TO THE WORLD
5005  DATA  40
5010  DATA  85,4,90,3,101,1,113,6,127,2,135,4,152,4,170,6,113,2,101,6
5020  DATA  AULD LANG SYNE
5050  DATA  40
5070  DATA  191,2,143,3,143,1,143,2,113,2,127,3,143,1,127,2,113,2
5072  DATA  143,3
6000  DATA  TWINKLE TWINKLE LITTLE STAR
6005  DATA  65
6010  DATA 127,1,127,1,85,1,85,1,76,1,76,1,85,2,95,1,95,1,101,1
6040  DATA  HOME ON THE RANGE
6042  DATA  35
6050  DATA  170,2,170,2,127,2,113,2,101,4,127,1,135,1,152,2,95,2,95,2
6080  DATA  BAA BAA BLACK SHEEP
6085  DATA  40
6090  DATA  170,2,170,2,113,2,113,2,101,1,101,1,101,1,101,1,
      113,4,127,2
7000  DATA  SAILING
7005  DATA  40
7010  DATA  127,3,95,3,95,3,127,3,113,1,120,1,113,1,95,2,113,1,127,5
7030  DATA  LONDON BRIDGE
7040  DATA  40
7050  DATA  95,3,85,1,95,2,107,2,113,2,107,2,95,4,127,2,113,2,107,4
```

**The program works in this way:**

Line 5: Enters the machine-language music routine into memory.

Line 6: Specifies the total number of tones played for each song.

Line 10: Gives the name of the program.

Line 11: Determines how many people will be playing.

Lines 12–16: Specify the number of songs in the program and dimensions a string array to reflect the number entered. (If you later decide to add more songs to this program, you must also change this number.)

Line 18: Reads the data in line 4000 into a string array. This particular program lists 7 song titles in line 4000. If you add more songs to the program, you must add the titles to this data line.

Line 20: Chooses a number at random from 1 through 7.

Line 30: Makes the variable C$ equal to the title contained in the cell number of the string array that corresponds to the number chosen in line 20.

Lines 35–38: Explain the rules of the game and determine if one or two players will participate.

Line 40: Reads the first line of data after the list of titles, which in this program is line 5000.
The data here is a song title. If the title is not the same as the one chosen at random, the next 21 items of data are read. These deal with notes and their relative duration. (See explanation for lines 5000–7050.)
The next song title is then read.

Line 50: Clears the keyboard buffer and reads in a number by which the relative duration of each note will be multiplied. Increasing or decreasing this number will change the speed at which the song is played.

Line 51: Decreases the number of tones left to be played. When this number reaches 0, the speaker is clicked, and a buzzing sound is produced to indicate that no more guesses may be made.

Line 53: If there are tones left to be played, the tone and its duration are read, and the machine-language music routine is called.

Line 55: Checks the keyboard buffer to see if a key has been pressed. If it has not been, then the program goes back to line 51, and another note is played.

Lines 57–70: Determine which player pressed a key to stop the song, and request that an answer be entered.

Lines 100–110: Give the correct answer and the score. All variables are zeroed out if the player decides to play again.

Line 111: Reads through the machine language data and song titles so that when the program is run again, beginning at line 20, the first item of data read will be in line 5000. The value of W should be changed if songs are added to or deleted from the program.

Lines 200–275: Determine if answers entered are correct.

Lines 280–300: Print the score for each player.

Lines 3000: Data for machine-language music routine.

Lines 4000: All the song titles contained in the program. These must be entered here in the same order that they are entered in data lines 5000–7050.

Line 5000: Title of the first song.

Line 5005: Number by which the relative durations of notes are multiplied.

Line 5010: Data made up of each APPLE II tone in the first song, followed by the tone's relative duration. There are 20 numbers in total, reflecting 10 notes and 10 durations.

Lines 5020–7050: Repeat the sequence of data found in lines 5000–5010 for each additional song.

In the second program, titled *Munch-A-Dot,* a moving mouth goes around the screen gobbling up dots. If the mouth goes into a space in which there are no dots, a buzzer sounds, and the game ends.

```
1  DIM A(24,16)
2  FOR R = 1 TO 24: FOR J = 1 TO 16:A(R,J) = 1: NEXT : NEXT
10  HGR : POKE  - 16302,0
12 X = 5:Y = 4
15  HCOLOR= 3: FOR R = 14 TO 268 STEP 11:
    FOR J = 16 TO 191 STEP 11: HPLOT R,J:NEXT : NEXT
18  FOR W = 1 TO 8:S =  INT ( RND (1) * 7) + 1:T = 22 * S:
    HCOLOR= 7: HPLOT 3 + T,5 + T:A(S * 2,S * 2) = 2:
    HPLOT 4 + T,5 + T
19  NEXT
20  HCOLOR= 6
30 A = 56:B = 50
40  HCOLOR= 6: GOSUB 1000
41  POKE  - 16368,0: FOR W = 1 TO 100: NEXT W
42  GET K$:KB =  ASC (K$): HCOLOR= 0: GOSUB 1000: GOTO 53
45 LB =  PEEK ( - 16384): IF LB < 128 THEN  GOTO 51
47 KB = LB - 128
51  HCOLOR= 0: GOSUB 1000: GOSUB 2000
53  HCOLOR= 6: IF KB = 68 THEN A = A + 11:X = X + 1:
    GOSUB 1000: GOTO 62
54  IF KB = 83 THEN  END
55  IF KB = 65 THEN A = A - 11:X = X - 1: GOSUB 2000: GOTO 62
57  IF KB = 87 THEN B = B - 11:Y = Y - 1: GOSUB 1000: GOTO 62
59  IF KB = 88 THEN B = B + 11:Y = Y + 1: GOSUB 1000: GOTO 62
60  PRINT  CHR$ (7); CHR$ (7)
62  IF A(X,Y) <  > 1 THEN  FOR R = 1 TO 100:M =
    PEEK ( - 16336): NEXT : GOTO 5000
63 A(X,Y) = 0:T = T + 1
65  GOTO 45
1000  HPLOT A,B TO A + 10,B - 6 TO A + 10,B - 5 TO A + 7,B
      TO A + 10,B + 5 TO A  + 10,B + 6 TO A,B
1002 M =  PEEK ( - 16336)
1005  RETURN
2000  HPLOT A + 11,B TO A + 1,B - 6 TO A + 1,B - 5 TO A + 4,
      B TO A + 1,B + 5 TO A + 1,B + 6 TO A + 11,B
2005  RETURN
5000  HOME : VTAB 21: POKE  - 16368,0
5001  PRINT "SCORE =";T;" DOTS": PRINT :T = 0
5002  POKE  - 16301,0: PRINT "TO PLAY AGAIN PRESS RETURN";:
      PRINT CHR$(7):INPUT A$:GOTO 2
```

The mouth can move up, down, left and right, using keys A,W,D, and X. To stop the mouth from moving, press key S, which also ends the game.

It is important to use only the four keys specified. If any other key is pressed, the game will end.

Each line works as follows:

| | |
|---|---|
| Lines 1–2: | Set up an array 24 by 16 and place the number 1 in each cell. |
| Line 10: | Calls the hi-res graphics screen and eliminates the text lines on the bottom of the screen. |
| Line 12: | States what cell in the array of dots corresponds to the position on the screen that is now occupied by the moving mouth. |
| Line 15: | Draws a dot matrix. |
| Line 18: | Chooses 8 dots at random to be poison dots. If these are eaten, the game ends. |
| Line 30: | Specifies at which point on the screen the mouth is located. If you want to place the mouth in a different position at the beginning, you must change the A, or horizontal position, and the B, or vertical position. |
| Line 40: | Sets the color and goes to subroutine 1000, which prints the mouth. |
| Line 41: | Clears the keyboard buffer. |
| Line 42: | Waits for keyboard input to determine in which direction the mouth will initially move. |
| Lines 45–60: | Check to see which key is pressed and move the mouth accordingly. |
| Line 62: | Checks to see if something other than the number 1 is present in the cell of the dot array corresponding to the screen position of the mouth. If an 0 appears, it indicates that the dot has already been eaten or that the dot was a poisoned one. In either case, the game ends. |
| Line 63: | The cell in the array which is referenced by the screen position occupied by the mouth receives a value of 0. The total number of dots eaten increases by 1. |
| Lines 1000–1005: | Draw the mouth facing right and click the speaker to produce an eating sound. |
| Lines 2000–2005: | Draw the mouth facing left and click the speaker. |
| Lines 5000–5002: | Clear the keyboard buffer and restore 4 lines of text at the bottom of the screen. The score is printed, and the player is asked if he or she wants to play again. |

## Supplementary Practice
## MUSIC FOR YOUR APPLE II

Here are eight musical selections that can be programmed into your computer. Two numbers are given for each note:

**APPLE tone:** This is the first number. A complete list of APPLE tones is shown in Exhibit 1. The G below middle C (APPLE tone 255) is the lowest musical tone available on the APPLE. The G located 1½ octaves above middle C (63) is the highest available musical tone.

If you want a rest between notes, input an APPLE tone of 0. Give it an appropriate relative time value.

**Relative time value:** This is the second number given for each note in the selections below. A note with a time value of 1 is the briefest note in a song. A note with a time value of 2 is twice as long as a note with a value of 1.

You have to decide the actual timing for each song. For a sprightly song, you may decide to have 1=30, 2=60, 3=90, etc. A slower song may require values of 1=50, 2=100, 3=150, etc.

Input the APPLE tone and time value for each of the first ten notes of a song. Then listen to how they sound. If the timing is too fast or too slow, adjust it, then listen to the new arrangement. Getting the time values to your liking early in the programming prevents your having to change the values for all the song's notes.

Toward the end of a song, you may wish to gradually increase time values slightly. You may even double the time value of the final note. This gives the song a more graceful ending.

Exhibit 5 shows the notes of the scale available to the APPLE II. Exhibit 6 shows a sample musical translation to APPLE tones.

The following program demonstrates how to program music into the APPLE II.

Enter and **RUN** it.

```
10  REM  PLAYING MUSIC ON THE APPLE
20  REM  LOADING IN MACHINE LANGUAGE MUSIC ROUTINE
30  FOR X = 768 TO 786: READ A: POKE X,A: NEXT X
40  REM  ROUTINE FOR READING AND PLAYING NOTES
50  READ N: REM  NUMBER OF NOTES
55  FOR X = 1 TO N
60  READ A,B: REM  APPLE TONE AND RELATIVE DURATION
64 S = 50
65 D = B * S: REM    SETTING THE DURATION
80  POKE 6,A: POKE 7,D: REM  PLACE NOTES AND DURATION IN MEMORY
90  CALL 768: REM  ENABLE MUSIC ROUTINE
100  NEXT X
110  END
```

```
500  REM  DATA FOR MACHINE LANGUAGE ROUTINE
510  DATA   173,48,192,136,208,4,198,7,240,8,202,208
515  DATA  246,166,6,76,0,3,96
600  REM  NUMBER OF NOTES
610  DATA  37
640  REM  NOTES AND DURATION
650  DATA  170,2,170,2,113,2,113,2,101,1,101,1,101,1,101
651  DATA  1,113,4,127,2,127,2,135,2,135,2,152,2
652  DATA  152,2,170,4,170,2,170,1,170,1,113,2,113,1,113
655  DATA  1,101,2,101,1,101,1,113,3,113,1,127,2,127,1,127
660  DATA  1,135,1,135,1,135,1,135,1,152,1,152,1,170,4
```

Following are several musical selections with APPLE tone and time values given. Experiment by entering and **RUN**ning them.

### Baa, Baa Black Sheep

| NOTE # | APPLE TONE | TIME VALUE | NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|---|---|---|
| 1 | 170 | 2 | 29 | 127 | 1 |
| 2 | 170 | 2 | 30 | 127 | 1 |
| 3 | 113 | 2 | 31 | 135 | 1 |
| 4 | 113 | 2 | 32 | 135 | 1 |
| 5 | 101 | 1 | 33 | 135 | 1 |
| 6 | 101 | 1 | 34 | 135 | 1 |
| 7 | 101 | 1 | 35 | 152 | 2 |
| 8 | 101 | 1 | 36 | 152 | 1 |
| 9 | 113 | 4 | 37 | 152 | 1 |
| 10 | 127 | 2 | 38 | 170 | 4 |
| 11 | 127 | 2 | | | |
| 12 | 135 | 2 | | | |
| 13 | 135 | 2 | | | |
| 14 | 152 | 2 | | | |
| 15 | 152 | 2 | | | |
| 16 | 170 | 4 | | | |
| 17 | 170 | 2 | | | |
| 18 | 170 | 1 | | | |
| 19 | 170 | 1 | | | |
| 20 | 113 | 2 | | | |
| 21 | 113 | 1 | | | |
| 22 | 113 | 1 | | | |
| 23 | 101 | 2 | | | |
| 24 | 101 | 1 | | | |
| 25 | 101 | 1 | | | |
| 26 | 113 | 3 | | | |
| 27 | 113 | 1 | | | |
| 28 | 127 | 2 | | | |

## Beethoven's Minuet in G

| NOTE # | APPLE TONE | TIME VALUE | NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|---|---|---|
| 1 | 101 | 3 | 41 | 63 | 8 |
| 2 | 95 | 1 | 42 | 63 | 1 |
| 3 | 85 | 3 | 43 | 67 | 1 |
| 4 | 90 | 1 | 44 | 76 | 1 |
| 5 | 85 | 3 | 45 | 85 | 1 |
| 6 | 90 | 1 | 46 | 95 | 4 |
| 7 | 85 | 3 | 47 | 101 | 4 |
| 8 | 90 | 1 | 48 | 76 | 3 |
| 9 | 85 | 8 | 49 | 95 | 1 |
| 10 | 76 | 3 | 50 | 101 | 4 |
| 11 | 101 | 1 | 51 | 113 | 2 |
| 12 | 95 | 8 | 52 | 0 | 2 |
| 13 | 85 | 3 | 53 | 127 | 3 |
| 14 | 113 | 1 | 54 | 113 | 1 |
| 15 | 101 | 4 | 55 | 101 | 3 |
| 16 | 0 | 4 | 56 | 107 | 1 |
| 17 | 127 | 3 | 57 | 101 | 3 |
| 18 | 113 | 1 | 58 | 107 | 1 |
| 19 | 101 | 3 | 59 | 101 | 3 |
| 20 | 107 | 1 | 60 | 107 | 1 |
| 21 | 101 | 3 | 61 | 101 | 8 |
| 22 | 107 | 1 | 62 | 95 | 3 |
| 23 | 101 | 3 | 63 | 120 | 1 |
| 24 | 107 | 1 | 64 | 113 | 8 |
| 25 | 101 | 8 | 65 | 101 | 3 |
| 26 | 113 | 2 | 66 | 135 | 1 |
| 27 | 127 | 2 | 67 | 127 | 4 |
| 28 | 127 | 2 | 68 | 0 | 2 |
| 29 | 135 | 2 | 69 | 85 | 2 |
| 30 | 135 | 2 | 70 | 90 | 2 |
| 31 | 113 | 2 | 71 | 85 | 2 |
| 32 | 127 | 2 | 72 | 101 | 2 |
| 33 | 152 | 2 | 73 | 85 | 2 |
| 34 | 170 | 4 | 74 | 127 | 2 |
| 35 | 0 | 4 | 75 | 101 | 2 |
| 36 | 85 | 2 | 76 | 170 | 2 |
| 37 | 63 | 2 | 77 | 101 | 2 |
| 38 | 63 | 4 | 78 | 113 | 2 |
| 39 | 67 | 4 | 79 | 95 | 2 |
| 40 | 63 | 4 | 80 | 135 | 2 |

## Beethoven's Minuet in G

| NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|
| 81 | 113 | 2 |
| 82 | 170 | 2 |
| 83 | 135 | 2 |
| 84 | 127 | 2 |
| 85 | 135 | 2 |
| 86 | 127 | 2 |
| 87 | 113 | 2 |
| 88 | 101 | 2 |
| 89 | 95 | 2 |
| 90 | 85 | 2 |
| 91 | 90 | 2 |
| 92 | 85 | 2 |
| 93 | 76 | 2 |
| 94 | 85 | 2 |
| 95 | 95 | 2 |
| 96 | 101 | 2 |
| 97 | 107 | 2 |
| 98 | 101 | 2 |
| 99 | 95 | 2 |
| 100 | 101 | 2 |
| 101 | 113 | 2 |
| 102 | 127 | 2 |
| 103 | 101 | 2 |
| 104 | 113 | 2 |
| 105 | 127 | 2 |
| 106 | 135 | 2 |
| 107 | 113 | 2 |
| 108 | 152 | 2 |
| 109 | 135 | 2 |
| 110 | 127 | 2 |
| 111 | 152 | 2 |
| 112 | 180 | 2 |
| 113 | 227 | 2 |
| 114 | 170 | 4 |
| 115 | 0 | 2 |
| 116 | 85 | 2 |
| 117 | 90 | 2 |
| 118 | 85 | 2 |
| 119 | 76 | 2 |
| 120 | 95 | 2 |
| 121 | 113 | 2 |
| 122 | 101 | 2 |
| 123 | 113 | 2 |
| 124 | 101 | 2 |
| 125 | 95 | 2 |
| 126 | 113 | 2 |
| 127 | 135 | 2 |
| 128 | 85 | 2 |
| 129 | 90 | 2 |
| 130 | 85 | 2 |
| 131 | 76 | 2 |
| 132 | 95 | 2 |
| 133 | 113 | 2 |
| 134 | 101 | 2 |
| 135 | 113 | 2 |
| 136 | 101 | 2 |
| 137 | 95 | 2 |
| 138 | 113 | 2 |
| 139 | 135 | 2 |
| 140 | 85 | 2 |
| 141 | 90 | 2 |
| 142 | 85 | 2 |
| 143 | 101 | 2 |
| 144 | 85 | 2 |
| 145 | 127 | 2 |
| 146 | 101 | 2 |
| 147 | 170 | 2 |
| 148 | 63 | 2 |
| 149 | 76 | 2 |
| 150 | 63 | 2 |
| 151 | 95 | 2 |
| 152 | 76 | 2 |
| 153 | 113 | 2 |
| 154 | 95 | 2 |
| 155 | 143 | 2 |
| 156 | 113 | 2 |
| 157 | 170 | 2 |
| 158 | 152 | 2 |
| 159 | 143 | 2 |
| 160 | 135 | 2 |
| 161 | 113 | 4 |
| 162 | 127 | 2 |
| 163 | 0 | 2 |
| 164–230 | repeat 1–67 | notes |

## Home on the Range

| NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|
| 1 | 170 | 2 |
| 2 | 170 | 2 |
| 3 | 127 | 2 |
| 4 | 113 | 2 |
| 5 | 101 | 4 |
| 6 | 127 | 1 |
| 7 | 135 | 1 |
| 8 | 152 | 2 |
| 9 | 95 | 2 |
| 10 | 95 | 2 |
| 11 | 95 | 4 |
| 12 | 95 | 1 |
| 13 | 95 | 1 |
| 14 | 85 | 4 |
| 15 | 127 | 1 |
| 16 | 127 | 1 |
| 17 | 127 | 2 |
| 18 | 135 | 2 |
| 19 | 127 | 2 |
| 20 | 113 | 10 |
| 21 | 170 | 2 |
| 22 | 170 | 2 |
| 23 | 127 | 2 |
| 24 | 113 | 2 |
| 25 | 101 | 4 |
| 26 | 127 | 1 |
| 27 | 135 | 1 |
| 28 | 152 | 2 |
| 29 | 95 | 2 |
| 30 | 95 | 2 |
| 31 | 95 | 4 |
| 32 | 95 | 1 |
| 33 | 95 | 1 |
| 34 | 101 | 3 |
| 35 | 113 | 1 |
| 36 | 127 | 2 |
| 37 | 135 | 3 |
| 38 | 127 | 1 |
| 39 | 113 | 2 |
| 40 | 127 | 8 |
| 41 | 0 | 4 |
| 42 | 85 | 6 |
| 43 | 95 | 2 |
| 44 | 101 | 3 |
| 45 | 113 | 1 |
| 46 | 101 | 10 |
| 47 | 170 | 1 |
| 48 | 170 | 1 |
| 49 | 127 | 4 |
| 50 | 127 | 1 |
| 51 | 127 | 1 |
| 52 | 127 | 3 |
| 53 | 135 | 1 |
| 54 | 127 | 2 |
| 55 | 113 | 8 |
| 56 | 0 | 2 |
| 57 | 170 | 2 |
| 58 | 170 | 2 |
| 59 | 127 | 2 |
| 60 | 113 | 2 |
| 61 | 101 | 4 |
| 62 | 127 | 1 |
| 63 | 135 | 1 |
| 64 | 152 | 2 |
| 65 | 95 | 2 |
| 66 | 95 | 2 |
| 67 | 95 | 4 |
| 68 | 95 | 1 |
| 69 | 95 | 1 |
| 70 | 101 | 3 |
| 71 | 113 | 1 |
| 72 | 127 | 2 |
| 73 | 135 | 2 |
| 74 | 127 | 2 |
| 75 | 113 | 2 |
| 76 | 127 | 12 |

## Sailing

| NOTE # | APPLE TONE | TIME VALUE | NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|---|---|---|
| 1 | 127 | 3 | 28 | 127 | 3 |
| 2 | 95 | 3 | 29 | 95 | 3 |
| 3 | 95 | 3 | 30 | 95 | 3 |
| 4 | 127 | 3 | 31 | 127 | 3 |
| 5 | 113 | 1 | 32 | 113 | 1 |
| 6 | 120 | 1 | 33 | 120 | 1 |
| 7 | 113 | 1 | 34 | 113 | 1 |
| 8 | 95 | 2 | 35 | 95 | 2 |
| 9 | 113 | 1 | 36 | 113 | 1 |
| 10 | 127 | 5 | 37 | 127 | 5 |
| 11 | 127 | 1 | 38 | 127 | 1 |
| 12 | 143 | 1 | 39 | 113 | 1 |
| 13 | 152 | 1 | 40 | 135 | 1 |
| 14 | 143 | 1 | 41 | 113 | 1 |
| 15 | 127 | 2 | 42 | 101 | 1 |
| 16 | 143 | 1 | 43 | 113 | 1 |
| 17 | 152 | 2 | 44 | 101 | 1 |
| 18 | 127 | 1 | 45 | 95 | 2 |
| 19 | 95 | 2 | 46 | 95 | 1 |
| 20 | 95 | 1 | 47 | 85 | 2 |
| 21 | 101 | 1 | 48 | 85 | 1 |
| 22 | 113 | 1 | 49 | 76 | 2 |
| 23 | 101 | 1 | 50 | 95 | 1 |
| 24 | 95 | 2 | 51 | 85 | 2 |
| 25 | 113 | 1 | 52 | 95 | 1 |
| 26 | 85 | 5 | 53 | 95 | 6 |
| 27 | 0 | 1 | | | |

## London Bridge

| NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|
| 1 | 95 | 3 |
| 2 | 85 | 1 |
| 3 | 95 | 2 |
| 4 | 107 | 2 |
| 5 | 113 | 2 |
| 6 | 107 | 2 |
| 7 | 95 | 4 |
| 8 | 127 | 2 |
| 9 | 113 | 2 |
| 10 | 107 | 4 |
| 11 | 113 | 2 |
| 12 | 107 | 2 |
| 13 | 95 | 4 |
| 14 | 95 | 3 |
| 15 | 85 | 1 |
| 16 | 95 | 2 |
| 17 | 107 | 2 |
| 18 | 113 | 2 |
| 19 | 107 | 2 |
| 20 | 95 | 4 |
| 21 | 127 | 4 |
| 22 | 95 | 4 |
| 23 | 113 | 2 |
| 24 | 143 | 6 |

## Twinkle, Twinkle Little Star

| NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|
| 1 | 127 | 1 |
| 2 | 127 | 1 |
| 3 | 85 | 1 |
| 4 | 85 | 1 |
| 5 | 76 | 1 |
| 6 | 76 | 1 |
| 7 | 85 | 2 |
| 8 | 95 | 1 |
| 9 | 95 | 1 |
| 10 | 101 | 1 |
| 11 | 101 | 1 |
| 12 | 113 | 1 |
| 13 | 113 | 1 |
| 14 | 127 | 2 |
| 15 | 85 | 1 |
| 16 | 85 | 1 |
| 17 | 95 | 1 |
| 18 | 95 | 1 |
| 19 | 101 | 1 |
| 20 | 101 | 1 |
| 21 | 113 | 2 |
| 22 | 85 | 1 |
| 23 | 85 | 1 |
| 24 | 95 | 1 |
| 25 | 95 | 1 |
| 26 | 101 | 1 |
| 27 | 101 | 1 |
| 28 | 113 | 2 |
| 29–42 | repeat #1–14 | notes |

## Auld Lang Syne

| NOTE # | APPLE TONE | TIME VALUE | NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|---|---|---|
| 1 | 191 | 2 | 31 | 85 | 2 |
| 2 | 143 | 3 | 32 | 95 | 3 |
| 3 | 143 | 1 | 33 | 113 | 1 |
| 4 | 143 | 2 | 34 | 113 | 2 |
| 5 | 113 | 2 | 35 | 143 | 2 |
| 6 | 127 | 3 | 36 | 127 | 3 |
| 7 | 143 | 1 | 37 | 143 | 1 |
| 9 | 127 | 2 | 38 | 127 | 2 |
| 10 | 113 | 2 | 39 | 85 | 2 |
| 11 | 143 | 3 | 40 | 95 | 3 |
| 12 | 143 | 1 | 41 | 113 | 1 |
| 13 | 113 | 2 | 42 | 113 | 2 |
| 14 | 95 | 2 | 43 | 95 | 2 |
| 15 | 85 | 6 | 44 | 85 | 6 |
| 16 | 85 | 2 | 45 | 71 | 2 |
| 17 | 95 | 3 | 46 | 95 | 3 |
| 18 | 113 | 1 | 47 | 113 | 1 |
| 19 | 113 | 2 | 48 | 113 | 2 |
| 20 | 143 | 2 | 49 | 143 | 2 |
| 21 | 127 | 3 | 50 | 127 | 3 |
| 22 | 143 | 1 | 51 | 143 | 1 |
| 23 | 127 | 2 | 52 | 143 | 2 |
| 24 | 113 | 2 | 53 | 113 | 1 |
| 25 | 143 | 3 | 54 | 127 | 1 |
| 26 | 170 | 1 | 55 | 143 | 3 |
| 27 | 170 | 2 | 56 | 170 | 1 |
| 28 | 191 | 2 | 57 | 170 | 2 |
| 29 | 143 | 6 | 58 | 191 | 2 |
| 30 | 0 | 1 | 59 | 143 | 8 |

## Home on the Range

| NOTE # | APPLE TONE | TIME VALUE | NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|---|---|---|
| 1 | 85 | 4 | 30 | 135 | 2 |
| 2 | 90 | 3 | 31 | 135 | 2 |
| 3 | 101 | 1 | 32 | 135 | 2 |
| 4 | 113 | 6 | 33 | 135 | 2 |
| 5 | 127 | 2 | 34 | 135 | 2 |
| 6 | 135 | 4 | 35 | 135 | 1 |
| 7 | 152 | 4 | 36 | 127 | 1 |
| 8 | 170 | 6 | 37 | 113 | 6 |
| 9 | 113 | 2 | 38 | 127 | 1 |
| 10 | 101 | 6 | 39 | 135 | 1 |
| 11 | 101 | 2 | 40 | 152 | 2 |
| 12 | 90 | 6 | 41 | 152 | 2 |
| 13 | 90 | 2 | 42 | 152 | 2 |
| 14 | 85 | 14 | 43 | 152 | 1 |
| 15 | 85 | 2 | 44 | 135 | 1 |
| 16 | 85 | 2 | 45 | 127 | 6 |
| 17 | 90 | 2 | 46 | 135 | 1 |
| 18 | 101 | 2 | 47 | 152 | 1 |
| 19 | 113 | 2 | 48 | 170 | 2 |
| 20 | 113 | 3 | 49 | 85 | 4 |
| 21 | 127 | 1 | 50 | 101 | 2 |
| 22 | 135 | 2 | 51 | 113 | 3 |
| 23 | 85 | 2 | 52 | 127 | 1 |
| 24 | 85 | 2 | 53 | 135 | 2 |
| 25 | 90 | 2 | 54 | 127 | 2 |
| 26 | 101 | 2 | 55 | 135 | 4 |
| 27 | 113 | 2 | 56 | 152 | 4 |
| 28 | 113 | 3 | 57 | 170 | 8 |
| 29 | 127 | 1 | | | |

# Exhibits

## EXHIBIT 1

APPLE TONES

| Musical Note | Apple Value |
|---|---|
| G | 255 |
| G# | 241 |
| A | 227 |
| A# | 214 |
| B | 202 |
| middle C | 191 |
| C# | 180 |
| D | 170 |
| D# | 161 |
| E | 152 |
| F | 143 |
| F# | 135 |
| G | 127 |
| G# | 120 |
| A | 113 |
| A# | 107 |
| B | 101 |
| C | 95 |
| C# | 90 |
| D | 85 |
| D# | 80 |
| E | 76 |
| F | 71 |
| F# | 67 |
| G | 63 |

## EXHIBIT 2

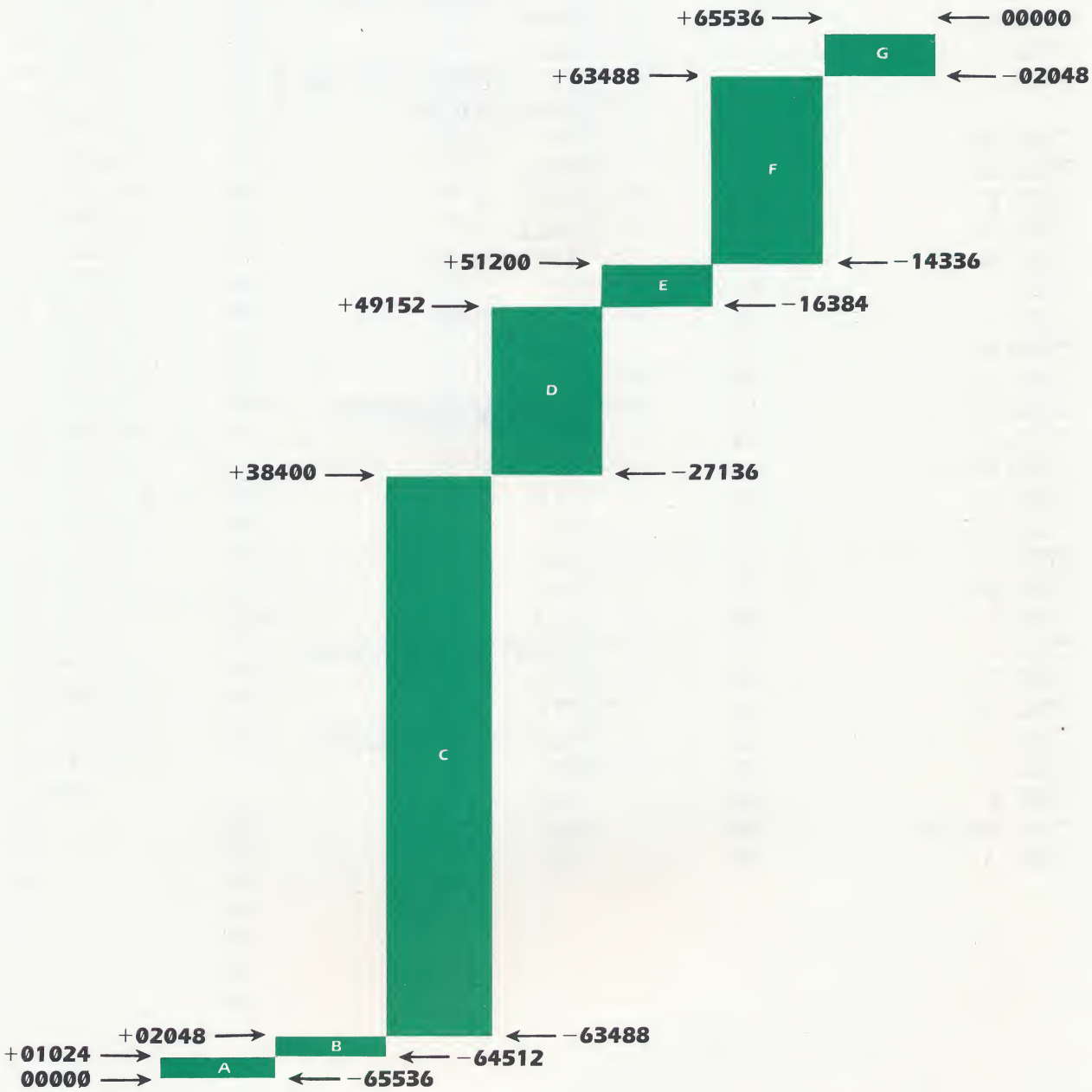
ASCII CODES FOR USE WITH **CHR$** STATEMENTS

| | | | |
|---|---|---|---|
| 0–CTRL–@ | 32–SPACE | 65–A | 96–′ |
| 1–CTRL–A | 33–! | 66–B | 97–a |
| 2–CTRL–B | 34–″ | 67–C | 98–b |
| 3–CTRL–C | 35–# | 68–D | 99–c |
| 4–CTRL–D | 36–$ | 69–E | 100–d |
| 5–CTRL–E | 37–% | 70–F | 101–e |
| 6–CTRL–F | 38–& | 71–G | 102–f |
| 7–CTRL–G | 39–’ | 72–H | 103–g |
| 8–CTRL–H | 40–( | 73–I | 104–h |
| 9–CTRL–I | 41–) | 74–J | 105–i |
| 10–CTRL–J | 42–* | 75–K | 106–j |
| 11–CTRL–K | 43–+ | 76–L | 107–k |
| 12–CTRL–L | 44–, | 77–M | 108–l |
| 13–CTRL–M | 45–- | 78–N | 109–m |
| 14–CTRL–N | 46–. | 79–O | 110–n |
| 15–CTRL–O | 47–/ | 80–P | 111–o |
| 16–CTRL–P | | 81–Q | 112–p |
| 17–CTRL–Q | 48–0 | 82–R | 113–q |
| 18–CTRL–R | 49–1 | 83–S | 114–r |
| 19–CTRL–S | 50–2 | 84–T | 115–s |
| 20–CTRL–T | 51–3 | 85–U | 116–t |
| 21–CTRL–U | 52–4 | 86–V | 117–u |
| 22–CTRL–V | 53–5 | 87–W | 118–v |
| 23–CTRL–W | 54–6 | 88–X | 119–w |
| 24–CTRL–X | 55–7 | 89–Y | 120–x |
| 25–CTRL–Y | 56–8 | 90–Z | 121–y |
| 26–CTRL–Z | 57–9 | 91–[ | 122–z |
| 27–ESC | | 92–\ | |
| 28– | 58–: | 93–] | |
| 29– | 59–; | 94–^ | |
| 30– | 60–< | 95–_ | |
| 31– | 61–= | | |
| | 62–> | | |
| | 63–? | | |
| | 64–@ | | |

**EXHIBIT 3**
CONTROL CHARACTERS

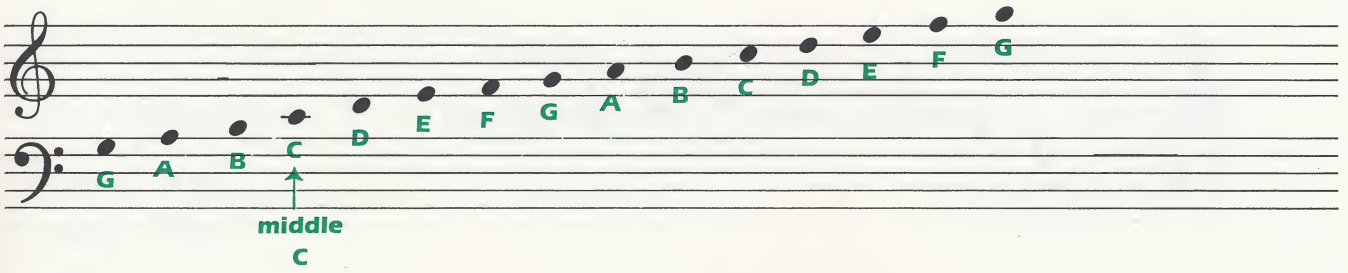
| CHARACTER | ASCII VALUE | USE |
|---|---|---|
| CTRL @ | 0 | NULL CHARACTER |
| CTRL A | 1 | SOH |
| CTRL B | 2 | STX |
| CTRL C | 3 | ETX (STOP PROGRAM EXECUTION) |
| CTRL D | 4 | ET |
| CTRL E | 5 | ENQ |
| CTRL F | 6 | ACK |
| CTRL G | 7 | BELL |
| CTRL H | 8 | BACKSPACE |
| CTRL I | 9 | HT |
| CTRL J | 10 | LINE FEED |
| CTRL K | 11 | VT |
| CTRL L | 12 | FF |
| CTRL M | 13 | CARRIAGE RETURN |
| CTRL N | 14 | SO |
| CTRL O | 15 | SI |
| CTRL P | 16 | DLE |
| CTRL Q | 17 | DC1 |
| CTRL R | 18 | DC2 |
| CTRL S | 19 | DC3 |
| CTRL T | 20 | DC4 |
| CTRL U | 21 | RIGHT ARROW (NAK) |
| CTRL V | 22 | SYN |
| CTRL W | 23 | ETB |
| CTRL X | 24 | CAN (CANCEL LINE) |
| CTRL Y | 25 | EM |
| CTRL Z | 26 | SUB |
| CTRL Shift M | 29 | GS |
| CTRL ^ | 30 | RS |

## EXHIBIT 4
## APPLE II MEMORY MAP




AREAS OF MEMORY MAP

**A**—∅
**B**—Text screen
**C**—Area available for program
**D**—DOS
**E**—I/O control
**F**—BASIC intrepreter (Applesoft)
**G**—System monitor program

## EXHIBIT 5

= 1/16 note

= 1/8 note = +

= 1/4 note = +

= 1/2 note = +

𝅝 = whole note = +

\# = sharp

♭ = flat

G# = A♭

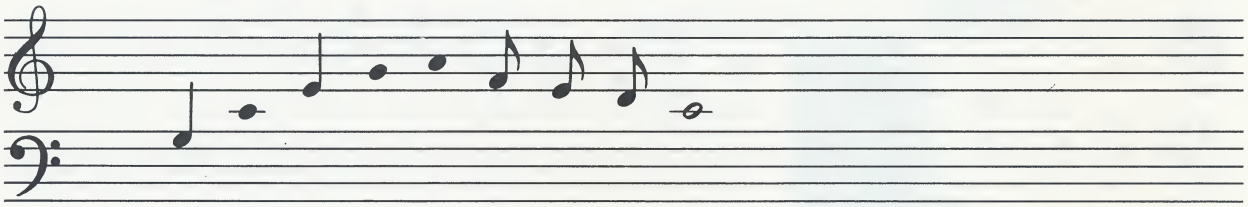
A# = B♭

B# = C

C# = D♭

etc.

## EXHIBIT 6



**First, find the note with the briefest duration. In this selection, it is an eighth note. Give the quarter note a time value of 1.**

| NOTE # | APPLE TONE | TIME VALUE |
|---|---|---|
| 1 | 255 | 2 |
| 2 | 191 | 4 |
| 3 | 152 | 2 |
| 4 | 127 | 4 |
| 5 | 113 | 4 |
| 6 | 143 | 1 |
| 7 | 152 | 1 |
| 8 | 170 | 1 |
| 9 | 191 | 8 |

## STEP BY STEP TWO PROGRESS CHART

**Name:** _____ **Date:** _____

| Lesson # | Date Completed | Need to Repeat | Quiz Score |
|---|---|---|---|
| Lesson 1, Part 1 | | | |
| Lesson 1, Part 2 | | | |
| Quiz 1 | | | |
| Lesson 2, Part 1 | | | |
| Lesson 2, Part 2 | | | |
| Quiz 2 | | | |
| Lesson 3, Part 1 | | | |
| Lesson 3, Part 2 | | | |
| Quiz 3 | | | |
| Lesson 4, Part 1 | | | |
| Lesson 4, Part 2 | | | |
| Quiz 4 | | | |
| Lesson 5, Part 1 | | | |
| Lesson 5, Part 2 | | | |
| Quiz 5 | | | |
| Introduction to Game Design | | | |
| Music For Your APPLE II | | | |

# STEP BY STEP TWO PROGRESS CHART

Name: _____ Date: _____

| Lesson # | Date Completed | Need to Repeat | Quiz Score |
|---|---|---|---|
| Lesson 1, Part 1 | | | |
| Lesson 1, Part 2 | | | |
| Quiz 1 | | | |
| Lesson 2, Part 1 | | | |
| Lesson 2, Part 2 | | | |
| Quiz 2 | | | |
| Lesson 3, Part 1 | | | |
| Lesson 3, Part 2 | | | |
| Quiz 3 | | | |
| Lesson 4, Part 1 | | | |
| Lesson 4, Part 2 | | | |
| Quiz 4 | | | |
| Lesson 5, Part 1 | | | |
| Lesson 5, Part 2 | | | |
| Quiz 5 | | | |
| Introduction to Game Design | | | |
| Music For Your APPLE II | | | |

Program Design, Inc. 95 East Putnam Avenue Greenwich, CT 06830